LITTLE MAN COMPUTER (CISC ARCHITECTURE SIMULATOR)

LMC APP PROGRAMMERS GUIDE

The app can be found at: https://app.craigndave.org/little_man_computer.html

URL PARAMETERS

URL parameters enable automatic configuration for your course:

`extended=false`	The model starts in basic mode (GCSE).
`extended=true`	The model starts in extended mode (A level).
`mbr=false`	The model shows a memory data register (OCR).
`mbr=true`	The model shows a memory buffer register (AQA).

E.g.

AQA GCSE:	https://app.craigndave.org/little_man_computer.html?extended=false&mbr=true
AQA A level:	https://app.craigndave.org/little_man_computer.html?extended=true&mbr=true
OCR GCSE:	https://app.craigndave.org/little_man_computer.html?extended=false&mbr=false
OCR A level:	https://app.craigndave.org/little_man_computer.html?extended=true&mbr=false

Please check your specification for the correct URL parameters.

NOTES

Instructions in green are only supported in the extended instruction set option.

INSTRUCTION SET

;

`; Comment` A comment. These are not assembled or executed.

ADD

`ADD 10` Adds the direct value stored at memory address 10 to the accumulator.

`ADD label` Adds the value stored in label to the accumulator. Defined with the DAT command.

`ADD #10` Adds the immediate value 10 to the accumulator.

`ADD +10` Adds the indexed value in IR+10 to the accumulator.

Supports +label.

`ADD @10` Adds the indirect value at address pointed to at address 10 to the accumulator.

Supports @label.

Alternative syntax: ADD [10]

Alternative syntax: ADD [label].

BRA

`BRA 10` Branch program to memory address 10.

`BRA label` Branch program to the memory address stored in label. Defined with the DAT command.

`BRA @10` Branch program to the indirect address pointed to at address 10.

Supports @label.

Alternative syntax: BRA [10]

Alternative syntax: BRA [label].

BRP

`BRP 10` Branch program to memory address 10 if accumulator is >= 0.

`BRP label` Branch program to the memory address stored in label if accumulator is >= 0. Defined with the DAT command.

`BRP @10` Branch program to the indirect address pointed to at address 10 if accumulator is >= 0.

Supports @label.

Alternative syntax: BRP [10]

Alternative syntax: BRP [label].

BRZ

`BRZ 10` Branch program to memory address 10 if accumulator is == 0.

`BRZ label` Branch program to the memory address stored in label if accumulator is == 0. Defined with the DAT command.

`BRZ @10` Branch program to the indirect address pointed to at address 10 if accumulator is == 0.

Supports @label.

Alternative syntax: BRZ [10]

Alternative syntax: BRZ [label].

DAT

`label DAT 10` Store the integer value 10 at an address identified as label.

`label DAT 'A'` Store a single character 'A' at an address identified as label. Note single quotes as the qualifier, not double quotes.

`label DAT [1, 2]` Store and array of integer values at an address identified as label and subsequent addresses, one for each character.

Supports ['A', 'B']

HLT

`HLT` Stops the program execution.

INP

`INP` Input a number and store in the accumulator.

Note this clears the input buffer first.

INX

`INX` Increment the index register (IR).

Equivalent to INX = INX + 1.

IPC

`IPC` Input a string if there are no characters in the input buffer. The first character of the input is removed and stored in the accumulator. The remainder of the string is held in the input buffer.

If there are one or more characters in the input buffer, remove the next character and store it in the accumulator.

LDA

`LDA 10`	Loads the direct value stored at memory address 10 to the accumulator.
`LDA label`	Loads the value stored in label to the accumulator. Defined with the DAT command.
`LDA #10`	Loads the immediate value 10 to the accumulator.
`LDA +10`	Loads the indexed value in IR+10 to the accumulator. Supports +label.
`LDA @10`	Loads the indirect value at address pointed to at address 10 to the accumulator. Supports @label. Alternative syntax: LDA [10] Alternative syntax: LDA [label].

LDX

`LDX`	Copy the value in the index register (IR) into the accumulator.

OTC

`OTC`	Output the value in the accumulator as an ASCII character.

OUT

`OUT`	Output the value in the accumulator.

STA

`STA 10`	Stores the value in the accumulator at the direct memory address 10.
`STA label`	Stores the value in the accumulator at the address of a label. Defined with the DAT command.
`STA +10`	Stores the value in the accumulator at indexed address in IR+10. Supports +label.
`STA @10`	Stores the value in the accumulator at the indirect address pointed to at address 10. Supports @label. Alternative syntax: STA [10] Alternative syntax: STA [label].

STO

Alternative command to STA for code compatibility with other Little Man Computer models.

STR

`label STR "Hi"`	Store a string "Hi". The first character is stored at an address identified as label and subsequent characters in subsequent addresses. Note double quotes as the qualifier, not single quotes.

STX

`STX`	Copy the value in the accumulator to the index register (IR).

SUB

`SUB 10`	Subtracts the direct value stored at memory address 10 from the accumulator.
`SUB label`	Subtracts the value of a label from the accumulator. Defined with the DAT command.
`SUB #10`	Subtracts the immediate value 10 from the accumulator.
`SUB +10`	Subtracts the indexed value in IR+10 from the accumulator.
	Supports +label.
`SUB @10`	Subtracts the indirect value at address pointed to at address 10 from the accumulator.
	Supports @label.
	Alternative syntax: SUB [10]
	Alternative syntax: SUB [label].