LMC – Little Man Computer

Instruction	Mnemonic	Info
Load	LDA xx	Load the contents of memory address xx onto the accumulator.
Store	STA xx	Store the contents of the accumulator to memory address xx.
Add	ADD xx	Add the contents of memory address xx to the accumulator.
Subtract	SUB xx	Subtract the contents memory address xx from the accumulator.
Input	INP	Copy the value from the "in box" onto the accumulator.
Output	OUT	Copy the value from the accumulator to the "out box".
End	HLT	Stops executing the program.
Data storage	DAT	Reserve as data the memory address reached when this instruction is compiled with an identifier e.g. num DAT. A value can be stored at the memory address by using DAT *value*
Branch always	BRA xx	Set the program counter to address xx. Basically, jump to another part of code.
Branch if zero	BRZ xx	If the accumulator is ZERO, set the program counter to address xx.
Branch if zero or positive	BRP xx	If the accumulator is ZERO or positive, set the program counter to address xx

Code example and explanation

```
1.          INP
2.          STA N1
3.          LDA N1
4.          ADD N1
5.          STA RES
6.          HLT
7.  N1      DAT
8.  RES     DAT
```

Line 1 gets an input from the user. Line 2 stores that number in the memory location reserved for the identifier 'N1'. Line 3 loads whatever is in the memory location reserved for N1 back into the accumulator. Line 4 adds whatever is in the accumulator with whatever is in the memory location labelled N1 (in this case adding the number to itself – doubling it). Line 5 stores the results, which are currently in the accumulator, in the memory location reserved for the identifier 'RES'. Line 6 end the program. Lines 7 and 8 are used to reserve memory locations after the instructions to be used in the program. We can use the identifiers in the code instead of the actual memory location number. When the code is compiled (after writing, before execution), the identifiers are swapped for the actual memory locations so that the CPU can find them.