LITTLE RISC COMPUTER (RISC ARCHITECTURE SIMULATOR)

LRC APP PROGRAMMERS GUIDE

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

NOTES

Usable registers are R0 to R11. R15 is nominally the program counter, but for compatibility this is labelled as the program counter.

Values are handled as JavaScript 32-bit signed integer results for ALU operations.

RAM has 100 locations by default, extended mode increases RAM to 1000 locations.

Values must be qualified with a #. E.g. #5 means the number 5.

Memory addresses are referred to either by their decimal address, e.g. 5 or by their label. Alternative syntax using square brackets for addresses is supported. E.g. [5] and [label].

Labels can be written either as `label:` or in the format `label COMMAND`

A colon label can appear on a line by itself and applies to the next instruction or data line

Multiple standalone labels can point at the same following instruction.

ALU FLAGS

The ALU maintains four flags: N, Z, C, V:

N	(negative)	Result is negative.
Z	(zero)	Result is zero.
C	(carry)	Unsigned carry on ADD, borrow on SUB, shifted-out bit on LSL/LSR.
V	(overflow)	Signed overflow on ADD/SUB.

INSTRUCTION SET

;

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

Alternative syntax of // to denote a comment is supported.

ADD

`ADD Rd, Rn, Ry` Adds the value in register y and register n, storing the result in register d.

`ADD Rd, Rn, #5` Adds the value 5 and register n, storing the result in register d.

Can set the N, Z, C, V flags.

AND

`AND Rd, Rn, Ry` Performs a bitwise logical AND operation between the value in register n and register y, storing the result in register d.

`AND Rd, Rn, #5` Performs a bitwise logical AND operation between the value 5 and the value in register n, storing the result in register d.

Can set the N, Z flags and clear the C, V flags.

B

`B 5` Branch program to memory address 5.

`B label` Branch program to a label.

BEQ

`BEQ 5` Branch program to memory address 5 if the Z (zero) flag is set.

`BRP label` Branch program to a label if the Z (zero) flag is set.

Use after the CMP command.

BGT

`BGT 5` Branch program to memory address 5 if the result of the previous operation was greater.

`BGT label` Branch program to a label if the result of the previous operation was greater.

Use after the CMP command.

BLT

`BLT 5` Branch program to memory address 5 if the result of the previous operation was smaller.

`BLT label` Branch program to a label if the result of the previous operation was smaller.

Use after the CMP command.

BNE

`BNE 5` Branch program to memory address 5 if the Z (zero) flag is not set.

`BNE label` Branch program to a label if the Z (zero) flag is not set.

Use after the CMP command.

BRA

Alternative command to B for code compatibility with other Little RISC Computer models.

CMP

`CMP Rn, Ry` Compare the value in register n with the value in register y.

`ADD Rn, #5` Compare the value in register n with the number 5.

Sets the N, Z, C, V flags.

Use before the BEQ, BGT, BLT commands.

DAT

`label DAT 10`	Store the integer value 10 at an address identified as label.
`label DAT label2`	Store the resolved address of label2.
`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']

EOR

`AND Rd, Rn, Ry`	Performs a bitwise logical XOR (exclusive OR) operation between the value in register n and register y, storing the result in register d.
`AND Rd, Rn, #5`	Performs a bitwise logical XOR (exclusive OR) operation between the value 5 and the value in register n, storing the result in register d.

Can set the N, Z flags and clear the C, V flags.

HALT

`HALT`	Stops the program execution.

INP

`INP Rd`	Input a number and store in register d. Note this clears the input buffer first.

LDR

`LDR Rd, 5` Loads the direct value stored at memory address 5 into register d.

`LDR Rd, label` Loads the value stored in label into register d. Defined with the DAT command.

Use MOV for loading immediate values.

LSL

`LSL Rd, Rn, #5` Logically shift left the value stored in register n by 5 and store the result in register d.

Sets the N, Z, C flags and clear the V flag.

LSR

`LSR Rd, Rn, #5` Logically shift right the value stored in register n by 5 and store the result in register d.

Sets the N, Z, C flags and clear the V flag.

MOV

`MOV Rd, Rn` Copy the value in register n into register d.

`MOV Rd, #5` Loads the value 5 into register d.

Can set the N, Z flags and clear the C, V flags.

MVN

`MOV Rd, Rn` Performs a bitwise logical NOT operation (inverting) register n, storing the result in register d.

`MOV Rd, #5` Performs a bitwise logical NOT operation (inverting) the value 5, storing the result in register d.

Can set the N, Z flags and clear the C, V flags.

NOT

Alternative command to MVN for code compatibility with other Little RISC Computer models.

OR

Alternative command to ORR for code compatibility with other Little RISC Computer models.

ORR

`ORR Rd, Rn, Ry`	Performs a bitwise logical OR operation between the value in register n and register y, storing the result in register d.
`ORR Rd, Rn, #5`	Performs a bitwise logical OR operation between the value 5 and the value in register n, storing the result in register d.

Can set the N, Z flags and clear the C, V flags.

OTC

`OTC Rd`	Output the value in register d as an ASCII character.
`OTC 10`	Outputs a newline.
`OTC x`	Outputs the ASCII character of x (32 to 128). Any other value outputs an underscore.

OUT

`OUT Rd`	Output the decimal value in register d followed by a newline.

STR

`STR Rd, 5`	Stores the value in register d into memory address 5.
`STR Rd, label`	Stores the value in register d into the memory address of a label. Defined with the DAT command.

SUB

`SUB Rd, Rn, Ry`	Subtracts the value in register y from register n, storing the result in register d.
`SUB Rd, Rn, #5`	Subtracts the value 5 from register n, storing the result in register d.

Can set the N, Z, C, V flags.

XOR

Alternative command to EOR for code compatibility with other Little RISC Computer models.
