Basic layout of any ARM program:

AREA <name>, CODE, READONLY

ENTRY

start

<code>

<more code>

...

SWI 0x11

END

Note: SWI 0x11 is not needed when using the Rolecks emulator.

Command layout:

**<space><COMMAND> <arg1>, <arg2>**<tab>; <comment>

The comment is optional.

Example:

MOV r0, #11 ; assign 11 to r0

MOV

Assign the value in the 2nd argument to the 1st argument.

Example:

MOV r0, #11

Assign the number 11 to register 0.

MOV r0, r1

Assign the value in register 1 to register 0.

MOV r0, #11

MVN

The same as MOV, but all the bits in the value assigned are inverted.

Example:

MVN r0, r1

Assign the number 11 to register 0.

If the number is in two’s complement, using MVN will get you the negative number one less than the one you moved.

Example:

MVN r0, #5

r0 now equals ‘–4’ in two’s complement.

ADD

Add the value in the 2nd argument to the 1st argument.

Example:

ADD r5, r1, r2

Add the number stored in register 2 to the number stored in register 1. Store the result in register 5.

SUB

Add the value in the 2nd argument to the 1st argument.

Example:

SUB r2, r0, #1

Subtract the number 1 from the number stored in register 0. Store the result in register 2.

AND

ORR

LDR

Loads the value at a main memory location into a register. Main memory is RAM.

Example:

LDR r0, #245624

Loads the value stored at main memory location 245624 into register 0.

Example:

LDR r0, r1

Uses the value stored in register 1 to find a main memory location. It then loads the value stored at that location in main memory into register 0.

STR

The opposite of LDR. Copies a value held in a register into main memory.

Example:

STR r0, #42

Stores the value held by register 0 in the main memory location 42.

CMP

Compares two values.

This operation takes r1 from r2. It then sets the CPSR depending on this calculation.

The Z bit is set to 1 if the two values are equal.

The N bit is set to 1 if r2 is greater than r1.

The Z and N bits are useful when branching. See BNE and BMI.

B

BNE

Only works if two numbers have been compared using CMP

A For Loop:

MOV r0, #0 Set up a counter.

start Set up a label to jump back to.

*<more code>* Do stuff.

ADD r0, r0, #1 Add 1 to whatever is stored in our counter, r0. Store the result in r0.

CMP r0, #10 Compare the counter to a number. This number is the total number of times the for loop will run. In this case, the loop will run 10 times.

BNE start Check the result of the comparison. If the counter and the total are not equal, the Z bit will not be set: so we are not finished looping. The BNE makes the program jump back to the ‘start’ label.

*<more code>* If the two numbers were equal, we are done looping. The program continues with the next step without jumping.

An If Statement, also known as a ‘Branch’:

CMP r0, r1 Compare r0 to r1.

BMI r0, r1 If r0 is less than r1, the N bit will be set to 1. If N is set to 1, the BMI makes the program jump to the ‘extra’ label.

B continue If r0 is greater than or equal to r1, then the jump doesn’t happen. Instead, the B command jumps the program directly to ‘continue’, missing out ‘extra’.

extra This is the ‘extra label.

*<do stuff>* Extra stuff is done…

B continue B makes the program jump to the ‘continue’ label.

continue This is the ‘continue’ label. The program continues as normal.

*<more code>*