Logical Operations:

AND{Cond}{S} desReg, op1, op2
Performs AND operation as desReg = op1 • op2

ORR{Cond}{S} desReg, op1, op2
Performs OR operation as desReg = op1 + op2

MVN{Cond}{S} desReg, op1
Performs NOT operation as desReg = (op1)'

EOR{Cond}{S} desReg, op1, op2
Performs XOR operation as desReg = op1 ⊕ op2

Some Useful Bit Manipulation

1. **To Clear Bits**
   Reset some bits to 0 using AND operation.
   Observe (0 • x = 0) and (1 • x = x)

2. **To Set Bits**
   Reset some bits to 1 using ORR operation.
   Observe (0 + x = x) and (1 + x = 1)

3. **To Invert Bits**
   Invert the value of some bits (1<==>0) using EOR operation.
   Observe (0 ⊕ x = x) and (1 ⊕ x = x')

Truth Tables (a reminder)

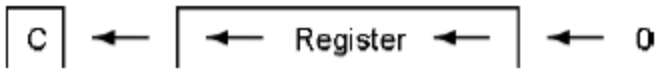
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A EOR B
0	0	0
0	1	1
1	0	1
1	1	0

A	NOT A
0	1
1	0

Logical Shift Left (LSL)



This will take the value of a register and shift the value up, towards the most significant bit, by n bits. The number of bits to shift is specified by either a constant value or another register. The lower bits of the value are replaced with a zero. This is a simple way of performing a multiply by a power of 2 ($x2^n$).

MOV R0, R1, LSL #2	R0 will become the value of R1 shifted left by 2. The value of R1 is unchanged.
MOV R0, R1, LSL R2	R0 will become the value of R1 shifted left by the number of bits specified in R2 register. R0 is the only register to change, both R1 and R2 are not affected by the operation.

If the instruction is to set the status register, the carry flag (C) is the last bit that was shifted out of the value.

Logical Shift Right (LSR)

Logical Shift Right is very similar to Logical Shift Left except it will shift the value to the right, towards the least significant bit, by n bits. It will replace the upper bits with zeros, thus providing an efficient unsigned divide by 2^n function ($| \div 2^n |$). The number of bits to shift may be specified by either a constant value or another register.

MOV R0, R1, LSR #2

R0 will take on the value of R1 shifted to the right by 2 bits. The value of R1 is not changed.

MOV R0, R1, LSR R2

R0 will become the value of R1 shifted to the right by the number of bits specified in the R2 register. R1 and R2 are not changed by this operation.

Arithmetic Shift Right (ASR)

The Arithmetic Shift Right is rather similar to the Logical Shift Right, but rather than replacing the upper bits with a zero, it maintains the value of the most significant bit. As the most significant bit is used to hold the sign, this means the sign of the value is maintained, thus providing a signed divide by 2^n operation ($\div 2^n$).

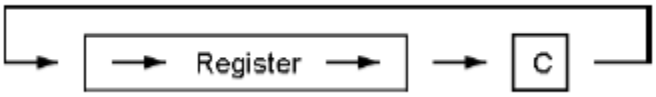
MOV R0, R1, ASR #2

Register R0 will become the value of register R1 shifted to the right by 2 bits, with the sign maintained.

MOV R0, R1, ASR R2

Register R0 will become the value of the register R1 shifted to the right by the number of bits specified by the R2 register. R1 and R2 are not changed by this operation.

Rotate Right



In the Rotate Right operation, the least significant bit is copied into the carry (C) flag, while the value of the C flag is copied into the most significant bit of the value. In this way none of the bits in the value are lost, but are simply moved from the lower bits to the upper bits of the value.

MOV R0, R1, ROR #2	This will rotate the value of R1 by two bits, and store the result in R0.
MOV R0, R1, ROR R2	Register R0 will become the value of the register R1 rotated to the right by the number of bits specified by the R2 register. R1 and R2 are not changed by this operation.