`MOV R0, R1, LSL #2`	R*0* will become the value of R*1* shifted left by 2 bits. The value of R*1* is not changed.
`MOV R0, R1, LSL R2`	R*0* will become the value of R*1* shifted left by the number of bits specified in the R*2* register. R*0* is the only register to change, both R*1* and R*2* are not effected by this 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

Logical Shift Right is very similar to Logical Shift Left except it will shift the value to the right, towards the lest 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`	R*0* will take on the value of R*1* shifted to the right by 2 bits. The value of R*1* is not changed.
`MOV R0, R1, LSR R2`	As before R*0* will become the value of R*1* shifted to the right by the number of bits specified in the R*2* register. R*1* and R*2* are not altered by this operation.

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

Arithmetic Shift Right

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 R*0* will become the value of register R*1* shifted to the right by 2 bits, with the sign maintained.
`MOV R0, R1, ASR R2`	Register R*0* will become the value of the register R*1* shifted to the right by the number of bits specified by the R*2* register. R*1* and R*2* are not altered by this operation.

Given the distinction between the Logical and Arithmetic Shift Right, why is there no Arithmetic Shift Left operation?

As a signed number is stored in two's complement the upper most bits hold the sign of the number. These bits can be considered insignificant unless the number is of a sufficient size to require their use. Thus an Arithmetic Shift Left is not required as the sign is automatically preserved by the Logical Shift.

Rotate Right

In the Rotate Right operation, the lest 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.