`LDR R0, [R1, #4]!`	Will load the register R*0* with the word at the memory address calculated by adding the constant value 4 to the memory address contained in the R*1* register. The new memory address is placed back into the base register, register R*1*.
`LDR R0, [R1, R2]!`	Loads the register R*0* with the value at the memory address calculated by adding the value in the register R*1* to the value held in the register R*2*. The offset register, R*2*, is not altered by this operation, the register holding the base address, R*1*, is modified to hold the new address.
`LDR R0, [R1, R2, LSL #2]!`	First calculates the new address by adding the value in the base address register, R*1*, to the value obtained by shifting the value in the offset register, R*2*, left by 2 bits. It will then load the 32-bit at this address into the destination register, R*0*. The new address is also written back into the base register, R*1*. The offset register, R*2*, will not be effected by this operation.

Post-Index Addressing

In *post-index address* the memory address is the base register value. As a side-effect, an offset is added to or subtracted from the base register value and the result is written back to the base register.

Post-index addressing uses the value of the base register without modification. It then applies the modification to the address and writes the new address back into the base register. This can be used to automatically increment or decrement a memory pointer after it has been used, so it is pointing to the next location to be used.

As the instruction must preform a write-back we do not need to include an exclamation mark. Rather we move the closing bracket to include only the base register, as that is the register holding the memory address we are going to access.

`LDR R0, [R1], #4`	Will load the register R*0* with the word at the memory address contained in the base register, R*1*. It will then calculate the new value of R*1* by adding the constant value 4 to the current value of R*1*.
`LDR R0, [R1], R2`	Loads the register R*0* with the value at the memory address held in the base register, R*1*. It will then calculate the new value for the base register by adding the value in the offset register, R*2*, to the current value of the base register. The offset register, R*2*, is not altered by this operation.
`LDR R0, [R1], R2, LSL #2`	First loads the 32-bit value at the memory address contained in the base register, R*1*, into the destination register, R*0*. It will then calculate the new value for the base register by adding the current value to the value obtained by shifting the value in the offset register, R*2*, left by 2 bits. The offset register, R*2*, will not be effected by this operation.