Offset Addressing

In *offset addressing* the memory address is formed by adding (or subtracting) an offset to or from the value held in a base register.

`LDR R0, [R1]`	Will load the register R*0* with the 32-bit word at the memory address held in the register R*1*. In this instruction there is no offset specified, so an offset of zero is assumed. The value of R*1* is not changed in this instruction.
`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 register R*1* is not changed by this instruction.
`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*. Both R*1* and R*2* are not altered by this operation.
`LDR R0, [R1, R2, LSL #2]`	Will load the register R*0* with the 32-bit value at the memory address calculated by adding the value in the R*1* register to the value obtained by shifting the value in R*2* left by 2 bits. Both registers, R*1* and R*2* are not effected by this operation.

This is particularly useful for indexing into a complex data structure. The start of the data structure is held in a *base* register, R*1* in this case, and the offset to access a particular field within the structure is then added to the base address. Placing the offset in a register allows it to be calculated at run time rather than fixed. This allows for looping though a table.

A scaled value can also be used to access a particular item of a table, where the size of the item is a power of two. For example, to locate item 7 in a table of 32-bit values we need only shift the index value 6 left by 2 bits (6×2^2) to calculate the value we need to add as an offset to the start of the table held in a register, R*1* in our example. Remember that the computer count from zero, thus we use an index value of 6 rather than 7. A 32-bit number requires 4 bytes of storage which is 2^2, thus we only need a 2-bit left shift.

Pre-Index Addressing

In *pre-index addressing* the memory address if formed in the same way as for offset addressing. The address is not only used to access memory, but the base register is also modified to hold the new value. In the ARM system this is known as a *write-back* and is denoted by placing a exclamation mark after at the end of the ⟨*op2*⟩ code.

Pre-Index address can be particularly useful in a loop as it can be used to automatically increment or decrement a counter or memory pointer.