Direccionamiento directo a registro
El operando se encuentra en un registro.
En la instrucción simplemente se debe codificar en qué registro se encuentra el operando.
Ejemplo

- add rd, rs, rn
- sub rd, rs, rn

Direccionamiento inmediato
El operando está en la propia instrucción.
Ejemplo

- add rd, rs, #Offset3
- sub rd, rs, #Offset3

Direccionamiento relativo a registro con desplazamiento
La dirección efectiva se calcula sumando una constante al contenido de un registro de dirección, cuyo rango dependerá del tamaño del operando o el destino.
Ejemplo

- ldr rd, [rb, #Offset5]
- str rd, [rb, #Offset5]

Direccionamiento relativo al contador de programa
Este modo especifica la dirección efectiva del operando o de un salto como la suma del contenido del contador de programa y un desplazamiento codificado en la propia instrucción.
Ejemplo

- **ldr** r0,=etiqueta o análogas, donde etiqueta es una etiqueta que referencia una dirección de memoria en un programa, el ensamblador sustituye dicha instrucción por una instrucción equivalente con direccionamiento relativo a PC.

Direccionamiento relativo al puntero de pila
Se lo utiliza como registro de direccionamiento el puntero de pila (sp, r14). La dirección efectiva es la suma del contenido del puntero de pila y un desplazamiento de 10 bits codificado. El direccionamiento relativo al puntero de pila se utiliza únicamente en las instrucciones **ldr** y **str**, y de manera implícita en las instrucciones **push** y **pop** de acceso a la pila.

Direccionamiento indirecto a registro con registro de desplazamiento
La dirección efectiva del operando es una dirección de memoria que se obtiene sumando el contenido de dos registros.
Ejemplo

- ldr rd, [rb, ro]: carga en el registro rd el contenido de la palabra de memoria indicada por la suma de los registros rb y ro.
- str rd, [rb, ro]: almacena el contenido del registro rd en la palabra de memoria indicada por la suma de los registros rb y ro.

Modo	Ejemplo
Directo a registro	`add r0, r1, r2`
Inmediato	`add r0, r1, #4`
Indirecto con desp.	`ldr r0, [r7, #4]`
Relativo a PC	`ldr r0, [pc, #20]`
Relativo a SP	`ldr r0, [sp, #20]`
Indirecto con RD	`ldr r0, [r1, r3]`

.byte
La directiva ***.byte Value8*** sirve para inicializar un byte con el contenido Value8.

.equ
Para declarar una constante se utiliza la directiva ***.equ*** de la siguiente forma:
.equ Constant, Value

ldrb
Para cargar bytes

Ascii y asciz
La directiva **.ascii "cadena"** le indica al ensamblador que debe inicializar la memoria con los códigos UTF-8 de los caracteres que componen la cadena entrecomillada. Dichos códigos se almacenan en posiciones consecutivas de memoria. La directiva **.asciz "cadena"** también sirve para declarar cadenas, la diferencia entre **.ascii** y **.asciz** radica en que la última añade un byte a 0 después del último carácter de la cadena.

Vector
Es un tipo de datos formado por un conjunto de datos del mismo tipo almacenados de forma secuencial. Para poder trabajar con un vector es necesario conocer la dirección de memoria en la que comienza y su tamaño.

balign
La directiva **.balign N** le indica al ensamblador que el siguiente dato que vaya a reservarse o inicializarse, debe comenzar en una dirección de memoria múltiplo de N.

space
La directiva **.space N** se utiliza para reservar N bytes de memoria e inicializarlos a 0.