Formato de las instrucciones de ensamblador ARM

Vamos a introducir brevemente el formato de las instrucciones del lenguaje ensamblador ARM y su uso general. Es importante para nosotros entender cómo funciona el lenguaje ensamblador, cómo se conectan las instrucciones y parámetros entre sí y qué se puede lograr al combinar instrucciones y parámetros.

Como hemos mencionado, el lenguaje ensamblador se compone de instrucciones, que son los bloques de construcción principales. Las instrucciones ARM generalmente van seguidas de uno o dos operandos y generalmente usan la siguiente plantilla:

MNEMONICO{S}{condition} {Rd}, Operand1, Operand2

 ¿Que significa mnemónico? Es una palabra que sustituye a un código de operación (lenguaje de máquina),

Dada la flexibilidad del conjunto de instrucciones ARM, no todas las instrucciones usan todos los campos provistos en la plantilla. Sin embargo, el propósito de los campos en la plantilla se describe a continuación:

MNEMONICO - Nombre corto de la instrucción (mnemonic)

{S} - Un sufijo opcional. Si se agrega S al final del Mnemonico, el registro CPSR de flags es actualizado en base a los resultados de una operación

{condition} - Condición que debe suceder para que se ejecute una instrucción

{Rd} - Registro destino para almacenar los resultados de una instrucción

Operand1 - Primer operando. Puede ser un valor inmediato ó un registro

Operand2 - Segundo operando (flexible). Puede ser un valor inmediato ó un registro ó un registro con shift, ver ejemplo a continuación

Mientras que los campos MNEMONICO, S, Rd y Operand1 son sencillos, la **condición** y los campos **Operand2** requieren un poco más de aclaración.

El campo de condición está estrechamente relacionado con el valor del registro CPSR, o para ser precisos, valores de bits específicos dentro del registro. El siguiente ejemplo muestra como se incluye el campo condición en una instrucción:

```
   MOVLE R0, #5             - Mueve el número 5 al registro R0
                              únicamente si la condición LE (Less than or
Equal - menor o igual que) es satisfecha
```

El operando2 se llama operando flexible, porque podemos usarlo de varias formas: como valor inmediato (con un conjunto limitado de valores), registro o registro con shift. Por ejemplo, podemos usar estas expresiones como Operand2:

```
   #123                    - Valor inmediato (limitado a un rango
dependiendo del procesador).
   Rx                      - Registro x (por ejemplo R1, R2, R3 ...)
   Rx, ASR n               - Registro x con shift aritmético a derecha de n
bits (1 = n = 32)
   Rx, LSL n               - Registro x con shift lógico a izquierda de n
bits (0 = n = 31)
   Rx, ROR n               - Register x con rotación a derecha de n bits (1
= n = 31)
```

Como un ejemplo completo de cómo se ven las instrucciones con el operando 2, veamos la siguiente lista:

```
   ADD   R0, R1, R2         - Suma los contenidos de R1 (Operando 1) and R2
(Operando 2 en la forma de un registro) y
                              almacena el resultado en R0 (Rd)
   ADD   R0, R1, #2         - Suma el contenido de R1 (Operando 1) y el
valor 2 (Operando 2 en la forma de un
                              valor inmediato) y almacena el resultado en R0
(Rd)
```

```
   MOV   R0, R1, LSL #1     - Mueve a R0 (Rd) el contenido de R1 (Operando 2
en la forma de registro con shift lógico
                              a izquierda) shifteado a izquierda un bit. Por
ejemplo si R1 tiene el valor 2, este es shifteado
                              a izquierda un bit y se convierte en 4. El 4
luego movido a R0.
```

Volver