Instrucciones de salto (Branch)

Las instrucciones de salto pueden producir saltos incondicionales (b y bx) o saltos condicionales. Cuando saltamos a una etiqueta empleamos **b** o **bal (branch always)**, mientras que si queremos saltar a un registro lo hacemos con bx. La variante de registro bx la solemos usar como instrucción de retorno de subrutina, raramente tiene otros usos.

En los saltos condicionales añadimos dos o tres letras a la (**b/bx**), mediante las cuales condicionamos si se salta o no dependiendo del estado de los flags. Estas condiciones se pueden añadir a cualquier otra instrucción, aunque la mayoría de las veces lo que nos interesa es controlar el flujo del programa y así ejecutar o no un grupo de instrucciones dependiendo del resultado de una operación (reflejado en los flags).

A continuación se muestra el formato de la instrucción branch. El sufijo **cond** representa la condición para que el salto se ejecute y **label** es una etiqueta que indica la posición de una instrucción en el programa. Si la condición no se cumple, el salto no se ejecuta y la ejecución continua con la instrucción siguiente al salto.

B{cond} label

Si no se especifica una condición, la instrucción dirige el flujo del programa a la posición indicada por la etiqueta, es un salto **incondicional**.

La lista completa de condiciones es ésta:

- **EQ** (equal, igual). Cuando Z está activo (Z vale 1).
- **NEQ** (not equal, igual). Cuando Z está inactivo (Z vale 0).
- **MI** (minus, negativo). Cuando N está activo (N vale 1).
- **PL** (plus, positivo o cero). Cuando N está inactivo (N vale 0).
- **CS/HS** (carry set/higher or same, carry activo/mayor o igual). Cuando C está activo (C vale 1).
- **CC/LO** (carry clear/lower, carry inactivo/menor). Cuando C está inactivo (C vale 0).
- **VS** (overlow set, desbordamiento activo). Cuando V está activo (V vale 1).
- **VC** (overlow clear, desbordamiento inactivo). Cuando V está inactivo (V vale 0).
- **GT** (greater than, mayor en complemento a dos). Cuando Z está inactivo y N=V (Z vale 0, N vale V).
- **LT** (lower than, menor en complemento a dos). Cuando N!=V (N vale not V).
- **GE** (greater or equal, mayor o igual en complemento a dos). Cuando N=V (N vale V).
- **LE** (lower or equal, menor o igual en complemento a dos). Cuando Z está activo y N!=V (Z vale 1, N vale not V).
- **HI** (higher, mayor). Cuando C está activo y Z inactivo (C vale 1, Z vale 0).
- **LS** (lower or same, menor o igual). Cuando C está inactivo ó Z activo (C vale 0 ó Z vale 1).

Por ejemplo, la instrucción beq destino_salto producirá un salto a la instrucción indicada por la

etiqueta destino_salto si y sólo si el bit de estado cero está activo (Z=1), y en caso contrario (Z=0) no interrumpirá el flujo secuencial de instrucciones. Previo a un salto condicional, el registro de flags debe ser actualizado mediante alguna instrucción aritmética (adds, subs, cmp, . . .) o lógica (ands, orrs, tst, . . .). En la mayoría de los casos tenemos que añadir el sufijo s a una instrucción normal add, para forzar que la nueva instrucción adds actualice los flags.

Un aspecto muy peculiar de la arquitectura ARM es que las llamadas a subrutinas se hacen mediante un sencillo añadido a la instrucción de salto. La instrucción bl (también blx) hace una llamada a una subrutina, mediante un salto a la subrutina y escribiendo en el registro lr la dirección de la siguiente instrucción.

Ver Subrutinas