Modos de direccionamiento

Las distintas formas de acceder a los parámetros de una instrucción se llaman modos de direccionamiento.

En la arquitectura ARM los accesos a memoria se hacen mediante instrucciones específicas **ldr** y **str**. El resto de instrucciones toman operandos desde registros o valores inmediatos, sin excepciones. En este caso la arquitectura nos fuerza a que trabajemos de un modo determinado: primero cargamos los registros desde memoria, luego procesamos el valor de estos registros con el amplio abanico de instrucciones del ARM, para finalmente volcar los resultados desde registros a memoria. Existen otras arquitecturas como la Intel x86, donde las instrucciones de procesado nos permiten leer o escribir directamente de memoria. Ningún método es mejor que otro, todo es cuestión de diseño. Normalmente se opta por direccionamiento a memoria en instrucciones de procesado en arquitecturas con un número reducido de registros, donde se emplea la memoria para guardar temporalmente. En nuestro caso disponemos de suficientes registros, por lo que podemos hacer el procesamiento sin necesidad de interactuar con la memoria, lo que por otro lado también es más rápido.

Direccionamiento inmediato. El operando fuente es una constante, formando parte de la instrucción.

```
mov r0, #1          /* r0 <-- 1 decimal */
add r2, r3, #4
```

Direccionamiento a registro o modo registro. En este modo el operador se encuentra en el registro.

```
add r2, r3, r1
```

Direccionamiento relativo a registro con desplazamiento. En este modo la dirección efectiva del operando es una dirección de memoria que se obtiene sumando el contenido de un registro y un desplazamiento especificado en la propia instrucción. Ejemplo: (str: store register to memory. Store immediate offset)

```
str r2, [r3, #8]
```

Es la forma más sencilla y admite 4 variantes. Después del acceso a memoria ningún registro implicado en el cálculo de la dirección se modifica.

- [Rx, #+inmediato]
- [Rx, #-inmediato]

Simplemente añade (o sustrae) un valor inmediato al registro dado para calcular la dirección. Es muy útil para acceder a elementos fijos de un array, ya que el desplazamiento es constante. Por ejemplo si tenemos r1 apuntando a un array de enteros de 32 bits int a[] y queremos poner a 1 el elemento a[3], lo hacemos así:

```
mov r2, #1              /* r2 <- 1 */
str r2, [r1, #+ 12]     /* ( r1 + 12) <- r2 */
```

Direccionamiento relativo a registro con registro de desplazamiento. En este modo la dirección efectiva del operando es una dirección de memoria que se obtiene sumando el contenido de dos registros. Ejemplo:

```
            ldr rd, [rb, ro]
```

Direccionamiento en las instrucciones de salto incondicional y condicional (direccionamiento absoluto). En el modo salto incondicional se usaría la instrucción b etiqueta, donde etiqueta es la dirección del salto, en el caso de salto condicional se usaría un operador condicional seguido de la etiqueta bxx etiqueta.

Volver