ORGA1 - Modos de direccionamiento

Enunciado: Práctica 3 - Arquitectura del CPU / Ejercicio 7

Dados los siguientes valores de la memoria y del registro R0 de la arquitectura ORGA1, ¿qué valores cargan en el registro R1 las siguientes instrucciones?

a) `MOV R1, 0x20` (inmediato)

b) `MOV R1, [0x20]` (directo)

c) `MOV R1, [[0x20]]` (indirecto)

d) `MOV R1, R0` (registro)

e) `MOV R1, [R0]` (indirecto registro)

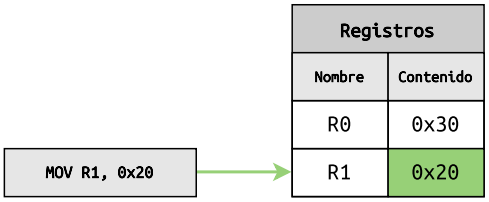
f) `MOV R1, [R0 + 0x20]` (indexado registro)

Memoria	
Dirección	Contenido
0x20	0x40
0x30	0x50
0x40	0x60
0x50	0x70

Registros	
Nombre	Contenido
R0	0x30
R1	???

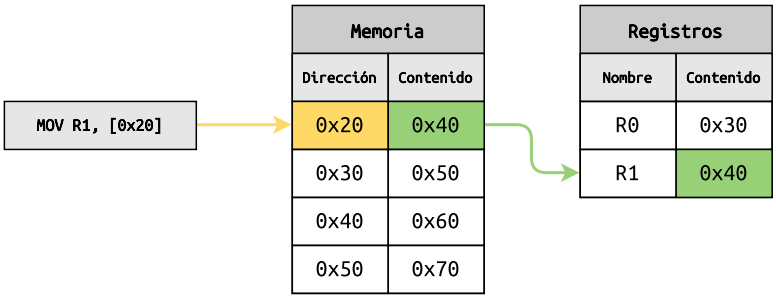
Resolución

a) `MOV R1, 0x20` **(inmediato)**



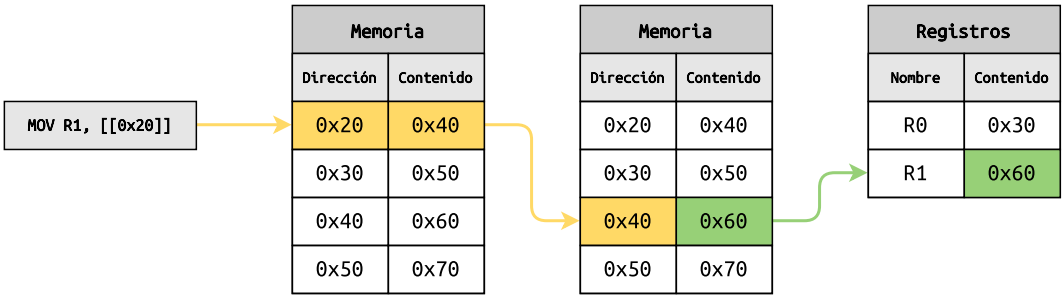
En este modo, el valor inmediato `0x20` incluído en la instrucción es directamente el valor que guardamos en R1. La nomenclatura "inmediato" hace referencia a que el valor está disponible inmediatamente como parte de la instrucción decodificada, no es necesario realizar ningún paso adicional.

b) `MOV R1, [0x20]` **(directo)**



En la sintaxis assembler de ORGA1, cuando nos encontramos un valor entre corchetes [] significa que debemos interpretarlo como una dirección de memoria. Para ejecutar la instrucción, primero leemos esa dirección de memoria y luego ese valor es el que termina usando la instrucción. En este caso, debemos leer la dirección `[0x20]` para obtener el valor `0x40`, el cual termina siendo guardado en R1.

c) `MOV R1, [[0x20]]` **(indirecto)**



Si el modo directo hace una traducción simple: dirección → contenido, el indirecto hace una traducción doble para obtener el operando de la instrucción: `[[0x20]]` → `[0x40]` → `0x60`. Necesitamos leer 2 veces la memoria para obtener el valor final.

Este modo es conceptualmente la misma idea que un puntero en C++. Un puntero es una variable que tiene su propia dirección de memoria, y en su contenido guardamos la dirección de algún otro dato que está en otro lugar de la memoria. En efecto, cuando desreferenciamos un puntero, estamos haciendo lo mismo que el modo de direccionamiento indirecto.

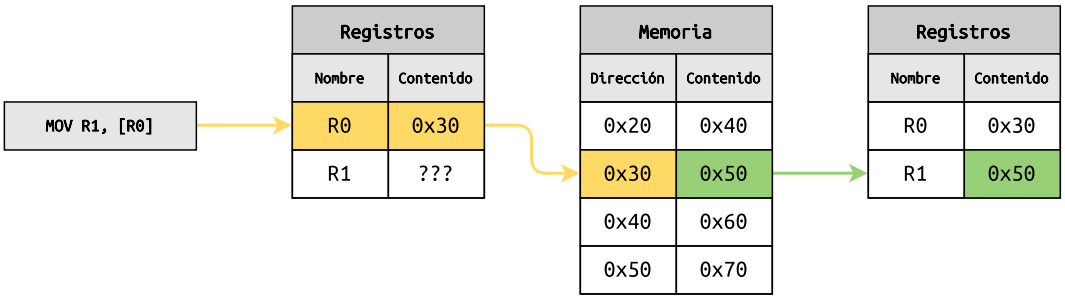
d) `MOV R1, R0` **(registro)**

Este modo es parecido al inmediato excepto que el valor a colocar en el registro destino lo obtenemos desde otro registro (en vez de especificarlo en la instrucción misma). Leemos `R0` y colocamos su valor `0x30` en R1.

¿El valor de R0 es parte de la instrución?

No, la instrucción únicamente codifica que queremos copiar el contenido de R0 en R1. A priori no sabemos qué valor estará guardado en R0 cuando se ejecute esta instrucción.

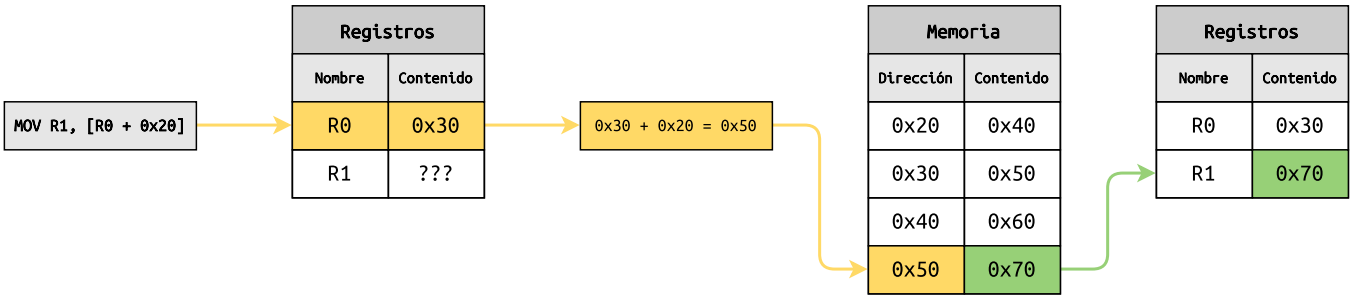
e) `MOV R1, [R0]` **(indirecto registro)**



Este modo es análogo al indirecto, solo que en vez de codificar la dirección como parte de la instrucción, interpretamos el valor guardado en R0 como una dirección: `[R0]` → `[0x30]` → `0x50`.

En la máquina ORGA1 sucede que la palabra y las direcciones de memoria ambas tienen 16 bits. Por lo tanto podemos usar el valor de un registro directamente como una dirección de memoria. Caso contrario sería necesario completar o recortar bits.

f) `MOV R1, [R0 + 0x20]` **(indexado registro)**



El modo indexado registro es como el modo registro, solo que utilizamos el valor guardado en el registro R0 como punto de partida y le sumamos un desplazamiento (el valor inmediato de la instrucción) para obtener la dirección final:
`[R0 + 0x20]` → `[0x30 + 0x20]` → `[0x50]` → `0x70`.