Uso de complemento a 2 en MCBE

El MCBE utiliza varias representaciones diferentes de datos numéricos, según la situación.

- Datos almacenados en la memoria o en el registro A
  - Se representan en **8 bits en complemento a 2**, ya que pueden ser negativos.
- Argumentos de instrucciones de transferencia de control (JMP, JZ)
  - Se representan en **5 bits en complemento a 2**, ya que son **desplazamientos** y pueden ser negativos.
- Argumentos de las demás instrucciones (LD, ST, ADD, SUB)
  - Se representan en **5 bits sin signo**, ya que son **direcciones** y no pueden ser negativos.
- Direcciones almacenadas en registros
  - Las direcciones almacenadas en el registro **PC** se representan en **8 bits sin signo**, aunque los tres más altos siempre son 0.

Instrucciones aritméticas

Para la suma o la resta **(ADD o SUB)**, se opera **en complemento a 2, en 8 bits**, entre el acumulador A y el contenido de la posición de memoria leída. El resultado queda en el acumulador.

Ejemplos

1) Supongamos que el MCBE encuentra la instrucción **10001011** en momentos en que el valor del acumulador es **00000011**. La instrucción es una **suma** (**100**) del contenido de la posición **01011** ($11_{(10)}$). Supongamos que en dicha posición existe un contenido **10101010**. Este dato está en complemento a 2, y como su bit de orden más alto es 1, es negativo. El resultado final del acumulador se obtiene sumando bit a bit el valor anterior más el dato. Tendremos **00000011 + 10101010 = 10101101**. ¿A qué valores decimales corresponden nuestros cálculos? ¿Ha ocurrido overflow?
2) Supongamos que se tiene la misma situación del caso anterior con la única diferencia de que la instrucción es **10101011**, y por lo tanto, es de **resta** (**101**) de la misma posición **01011** ($11_{(10)}$). Como antes, supongamos que en dicha posición existe un contenido **10101010**. Tendremos **A - Dato = 00000011 – 10101010**. Como se pide una **resta**, debemos **sumar el complemento del sustraendo**. Para esto complementamos 10101010 invirtiendo los bits → 01010101 y sumando 1 → 01010110. Tendremos A – Dato = A + C2(Dato) = **00000011 + 01010110 = 01011001**. Si convertimos todos los valores a decimal veremos que A contenía el valor **3**, y que la instrucción implicaba **restarle -86**, o sea, **sumarle 86**. El resultado es **89**. No ha habido overflow.
3) Supongamos la misma situación que en 1) pero ahora el valor de A es **10000001**. Hay que sumarle **10101010.** Como ambos números están en complemento a 2, podemos hacer la suma bit a bit. Encontraremos que **10000001 + 10101010 = (1)00101011**, donde el bit entre paréntesis representa el *carry out*. Como éste es diferente del *carry in* (que es **0**), tenemos

overflow, el bit de *carry out* se pierde, y en el registro A queda el número **00101011**. Este número es positivo, cuando debía ser negativo porque estábamos sumando 10000001 ($-127_{(10)}$) y 10101010 ($-86_{(10)}$). El resultado debe ser descartado.

Instrucciones de transferencia de control

Analicemos lo que ocurre cuando el MCBE acaba de traer una instrucción **de transferencia de control (JMP o JZ)** al IR y debe ejecutarla.

- El PC contiene una cierta dirección (aquella donde se halló la instrucción de transferencia de control). Esta dirección está expresada como entero sin signo en 8 bits. Los bits que expresan la dirección ocupan los cinco bits más bajos, y los tres más altos son 0.
- El desplazamiento, ya sea positivo o negativo, está almacenado en los cinco bits de orden inferior de la instrucción, en complemento a 2, y debe sumarse al PC.
- Ahora bien, para poder hacer esta suma **PC + desplazamiento**, tenemos que tener ambos números en **complemento a 2 en igual cantidad de bits**. Como el valor de la dirección es positivo, su expresión en complemento a 2 en 8 bits coincide con la expresión sin signo.
- Para representar el desplazamiento en 8 bits aplicamos la llamada **regla de extensión de signo**: completamos los 8 bits necesarios agregando 3 bits a la izquierda, que serán 0 si el desplazamiento es positivo, y 1 si es negativo. Luego, ambos números **ya están en complemento a 2** en ocho bits y pueden directamente sumarse, **simplemente por suma directa bit a bit**, vigilando, claro, si se presentan situaciones de **overflow.**

Ejemplo

Supongamos que el MCBE encuentra en la dirección **13** la instrucción **11011010**. Analicemos lo que ocurrirá.

- Esta instrucción es una transferencia de control incondicional (110). El desplazamiento está expresado en complemento a 2 en los cinco bits restantes (11010). Luego, por comenzar en 1, el desplazamiento es negativo. Al sumarle algebraicamente este desplazamiento al PC, su nuevo valor será menor que el actual: la transferencia de control se realizará a una dirección "más baja" que la apuntada actualmente por el PC (es decir, menor que 13).
- Como la instrucción fue encontrada en la dirección 13, el contenido actual del PC será **00001101**. Como el desplazamiento es **11010**, por extensión de signo obtendremos **11111010**. Sumando el PC con el desplazamiento: **00001101 + 11111010 = 00000111 = $7_{(10}$**. La siguiente instrucción a ejecutar será la que está en la dirección 7. Notemos que no ha habido overflow (porque los bits de *carry in* y de *carry out* son iguales).
- Si queremos conocer la cantidad de bytes que hay que "retroceder", sólo necesitamos averiguar el valor absoluto del desplazamiento. Lo hacemos complementando a 2:
  - Invertimos los bits de 11010 → 00101 y sumamos 1 → 00110. El valor absoluto del desplazamiento es 6, es decir, necesitamos retroceder 6 bytes.
  - Sin embargo, no es necesario averiguar este valor absoluto para realizar la operación de calcular la nueva dirección: directamente podemos sumar bit a bit ese desplazamiento al valor actual del PC, y utilizar el nuevo valor del PC en el siguiente ciclo de instrucción.