

Complementos: a 1 y 2

El complemento a 1 y a 2 de un número binario permiten la representación de números negativos. El método de complemento a 2 es comúnmente usado por las computadoras para representar los negativos.

Complemento a 1

El complemento a 1 de un número binario se obtiene representándolo primero en **SM**, es decir el bit más significativo, representa el signo y el resto de los bits la magnitud. Luego se procede de 2 formas distintas dependiendo del signo del valor a representar. El complemento a 1 $C_{(1)}$ de un valor es:

- **Si el número a representar es positivo:** la representación queda igual que en signo magnitud. Por ejemplo si tenemos el número positivo $6_{(10)}$ y debemos pasarlo a $C_{(1)}$, quedaría así:

0	1	1	0	(2)
↓	↓	↓	↓	
0	1	1	0	(c1)

- **Si el número a representar es negativo:** Se efectúa el complemento dígito a dígito de la representación de la magnitud obtenida anteriormente. Veamos como representar en $C_{(1)}$ el número $-6_{(10)}$:

Primero lo representamos en SM, como estamos usando 4 bits quedaría así

1	1	1	0		SM

Luego complementamos los bits restantes

1	0	0	1		C_1

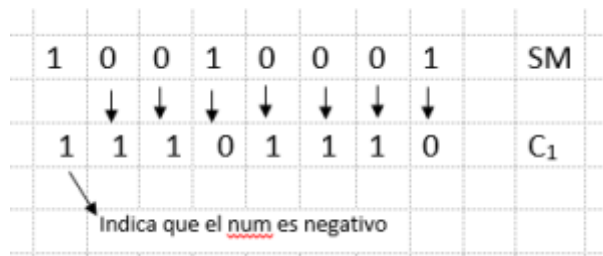
El bit mas significativo, nos indica que el número es negativo. **-6**.

Analicemos otro ejemplo; pero ahora con 8 bits (1 byte), representemos el número -17_{10} en C_1 .

Paso 0) Como primer paso escribimos el número en SM:

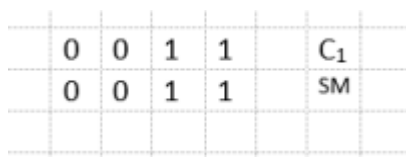
1	0	0	1	0	0	0	1		SM

Paso 1) Como el número a representar es negativo (**-17**), complemento y queda así:



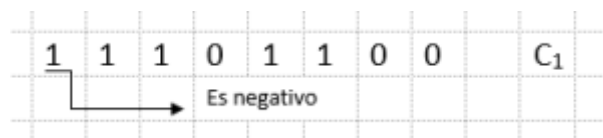
Ahora supongamos que tenemos un número que está en C₁: **0011** ¿Cómo lo pasamos a SM?.

Paso 0) Si el número tiene en su bit más significativo un **0**, quiere decir que es positivo, entonces su representación quedaría exactamente igual.

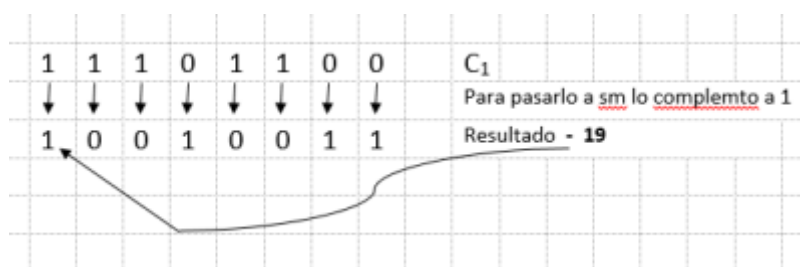


Probemos otro número que está en C₁: **11101100** ¿Cómo lo pasamos a SM o a binario?. **¡Lo volvemos a complementar a C₁!**

Paso 0) Si el número tiene en su bit más significativo un **0**, quiere decir que es positivo, entonces su representación quedaría exactamente igual. Este no es el caso ya el bit 7 tiene valor 1



Paso 1) Complemento para obtener el valor numérico en SM.



Pensemos ahora, en el rango de representación, dicho de otra forma, ¿Cuál es número más grande y el más chico que puedo representar con n bits?

Partamos de un ejemplo fácil con 4 bits

C ₁	Decimal
1000	-7
1001	-6
1010	-5
1011	-4

C_1	Decimal
1100	-3
1101	-2
1110	-1
1000	-0
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Vemos que el 0 tiene doble representación, lo cual genera un problema. Veamos el rango de representación, sería así: $[-(2^{4-1}-1), 2^{4-1}-1] \rightarrow (-7, 7)$

Generalizando para n bits:

$$[-(2^{n-1}-1), 2^{n-1}-1]$$

Complemento a 2

En la actualidad la representación más utilizada en las computadoras es complemento a dos. Sin embargo, primero estudiamos las otras representaciones dado que son más simples y sirven como una buena base para el complemento a dos.

Veamos mediante ejemplos como pasar un número binario SM a C_2

El complemento a 2 C_2 de un número binario se obtiene, primero lo pasamos a C_1 luego le sumamos 1, al bit menos significativo.

- **Si el número a representar es positivo:** la representación queda igual que en signo magnitud. Por ejemplo si tenemos el número positivo $6_{(10)}$ y debemos pasarlo a $C_{(2)}$, quedaría así:

0	1	1	0	sm	6
0	1	1	0	C_2	6

- **Si el número a representar es negativo:** Se efectúa el complemento dígito a dígito de la representación de la magnitud obtenida anteriormente. Veamos como representar en $C_{(2)}$ el número $-6_{(10)}$:

Primero lo representamos en **SM**, luego en C_1 y para convertirlo finalmente a C_2 , le sumamos 1 al bit menos significativo. Como estamos usando 4 bits quedaría así

1	1	1	0		<u>sm</u>	-6
	↓	↓	↓			
1	0	0	1		C1	-6

y ahora sumamos 1

			1			
	1	0	0	1		C1 -6
+				1		
	0	0	1	0		C2 -6

Sigamos con más ejemplos, convirtamos por ejemplo con el número $-74_{(10)}$ a C_2 trabajando con 8 bits:

- **Paso 0)** SM: Si el número tiene en su bit más significativo un **0**, quiere decir que es positivo, entonces su representación quedaría exactamente igual. Este no es el caso (**11001010 SM**) ya el bit 7 tiene valor 1

1	1	0	0	1	0	1	0	<u>sm</u>	-74
---	---	---	---	---	---	---	---	-----------	-----

- **Paso 1)** Convertimos a C_1

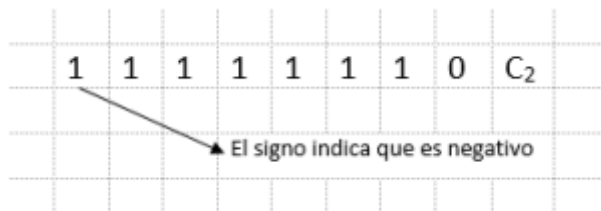
1	1	0	0	1	0	1	0	<u>sm</u>	-74
	↓	↓	↓	↓	↓	↓	↓		
1	0	1	1	0	1	0	1	C ₁	-74

- **Paso 2)** Convertimos a C_2 simplemente sumando 1 al bit menos significativo

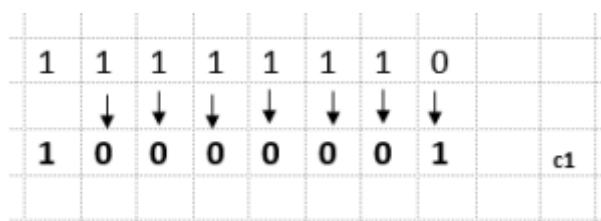
						1			
1	0	1	1	0	1	0	1	C ₁	-74
							1		
1	0	1	1	0	1	1	0	C ₂	-74

Ahora supongamos que tenemos un número que está en C_2 : **10110110** ¿Cómo lo pasamos a SM?.
¡Lo complementamos a 2!

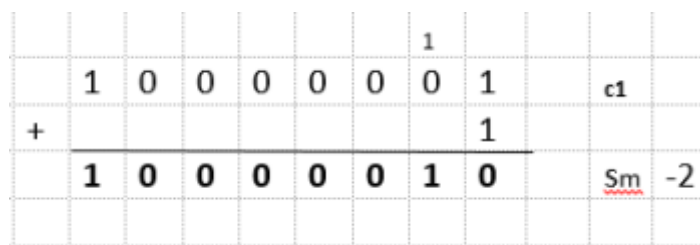
- Paso 0) Analizamos el signo del bit más significativo para saber si es negativo (1). Si no fuera así la representación en C_2 quedaría igual



- Paso 1) Complementamos C_1



- Paso 2) Sumamos 1 al bit menos significativo y nos queda el número en signo magnitud



Veamos el rango de representación, ¿Cuál es número más grande y el más chico que puedo representar con n bits?

Ejemplo fácil con 4 bits

C_2	Decimal
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Ahora vemos que el 0 no presenta el problema de la doble representación de C_1 . Veamos el rango de representación, sería así: $[-(2^{4-1}), 2^{4-1}-1] \rightarrow (-8, 7)$

Generalizando para n bits:

$[-(2^{n-1}), 2^{n-1}-1]$

Tabla Comparativa (representación en 4 bits)

Número decimal	Signo y magnitud	Complemento a 1	Complemento a 2
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
0	0000	0000	0000
-0	1000	1111	No existe
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	no existe	No existe	1000

	Binario puro	Signo Magnitud	Complemento a uno	Complemento a dos
Rango de representación	$[0;+2^n-1]$ Asimétrico	$[-(2^{n-1}-1);+2^{n-1}-1]$ Simétrico	$[-(2^{n-1}-1);+2^{n-1}-1]$ Simétrico	$[-(2^{n-1});+2^{n-1}-1]$ Asimétrico
Permite representar números con signo negativo (ventaja)	No	Si	Si	Si
Tiene cero positivo y negativo (desventaja)	No	Si	Si	No, pero nos permite representar un número más.
Facilita operaciones aritméticas (ventaja)	Si	No	Si	Si

Volver

— Mariano Vargas 

From:

<http://wiki.educabit.ar/> - **Wiki Sistemas**

Permanent link:

http://wiki.educabit.ar/doku.php?id=representacion_de_datos

Last update: **2025/09/11 22:48**

