1 Première tentative

L'ordinateur représente toutes les données sous formes de "**mots**" binaires de taille fixe (8, 16 32, 64 bits). La question se pose alors de savoir comment représenter les entiers relatifs sous cette forme de taille limitée (un octet pour représenter un entier par exemple).

Pour représenter les nombres relatifs (signe + ou signe -) en binaire, la première idée qui vient à l'esprit est d'utiliser un bit de signe avant de coder la valeur absolue du nombre (par exemple 0 pour les positifs, et 1 pour les négatifs).

On aurait ainsi pour un codage sur 8 bits (1 bit de signe et 7 bits pour la valeur absolue) :
$+23_{10} = 00010111_2$
$-23_{10} = 10010111_2$

Un inconvénient est que 0 peut se coder de 2 façons : 0000 0000 ou 1000 0000.

Mais le plus gênant est pour effectuer des additions par la suite :

Exemple :

			^{1}		^{1}	^{1}	^{1}		(retenues)
	0	0	0	1	0	1	1	1	(23)
+	1	0	0	1	0	1	1	1	(-23)
	1	0	1	0	1	1	1	0	(-46)

qui s'interprète comme $-(32+8+4+2)_{10} = -46 \cdots$ au lieu de $\cdots 0$!

2 Le complément à 2 (2^n)

2.1 Le complément à 10 (10^n)

Une solution originale s'est imposée. Voyons d'abord le principe en base décimale.
Pour représenter un nombre négatif en se limitant à 8 chiffres, on peut décider de le représenter par le complément à 10^8 (car 8 chiffres) de sa valeur absolue.

Exemple : le nombre -243 se représente alors par $1\ 0000\ 0000 - 243 = 9999\ 9757$
Que l'on peut aussi écrire : $9999\ 9999 - 243 + 1 = 9999\ 9756 + 1 = 9999\ 9757$
Pour avoir le complément à 10, on prend le complément à 9 de chaque chiffre, et on ajoute 1.

Avec cette représentation un nombre négatif commence toujours par 9.

La magie s'opère lorsqu'on fait l'addition de deux nombres, en se souvenant bien qu'on est limité à 8 chiffres : une éventuelle retenue de débordement disparaît.

Exemples :

(1)	1	1	1	1	1	1	1		(retenues)
	0	0	0	0	0	2	4	3	(243)
+	9	9	9	9	9	7	5	7	(-243)
(1)	0	0	0	0	0	0	0	0	(0)

(1)	1	1	1	1	1		1		(retenues)
	0	0	0	0	0	5	2	7	(527)
+	9	9	9	9	9	7	5	7	(-243)
(1)	0	0	0	0	0	2	8	4	(+284 : positif, commence par un 0)

							1		(retenues)
	0	0	0	0	0	1	2	7	(127)
+	9	9	9	9	9	7	5	7	(-243)
	9	9	9	9	9	8	8	4	(-116 : complément à 10^8, négatif, commence par un 9)

2.2 Le complément à 2 (2^n)

En binaire, on fait exactement la même chose avec le complément à 2^n, abrégé habituellement en "complément à 2", (n est le nombre de bits pour représenter le nombre).

La méthode pour coder un nombre négatif est donc de prendre le complément à 2 de sa valeur absolue, soit :

1. Prendre le complément à 1 de chaque bit, c'est à dire inverser les 0 et les 1 du nombre.
2. Ajouter 1.

Exemple : représenter le nombre $-(94)_{10}$ en binaire en complément à 2 :

1. Conversion de 94_{10} en binaire :
   $(94)_{10} = 64 + 16 + 8 + 4 + 2 = 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = (0101\ 1110)_2$
2. Complément à 1 : $0101\ 1110 \rightarrowtail 1010\ 0001$
3. On ajoute 1 : $1010\ 0001 + 1 = 1010\ 0010$

Finalement $-(94)_{10} \rightarrowtail 1010\ 0010$.

Exercice : Calculer la somme $-(94)_{10}$ et $+(33)_{10}$ en binaire en complément à 2 :

		1							(retenues)
	1	0	1	0	0	0	1	0	(-94)
+	0	0	1	0	0	0	0	1	(+33)
	1	1	0	0	0	0	1	1	(- ?)

Il reste à décoder le résultat en prenant son complément à 2.

1. Complément à 1 : $1100\ 0011 \rightarrowtail 0011\ 1100$
2. On ajoute 1 : $0011\ 1100 + 1 = 0011\ 1101$
3. Conversion en décimal :
   $(0011\ 1101)_2 = 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 32 + 16 + 8 + 4 + 1 = (61)_{10}$

Finalement on trouve bien $-61 = -94 + 33$.