Programmation 1: Fondamentaux

Binaire et Codage

- Représentation des entiers en binaire
- Opérations sur les entiers



Représentation des entiers en binaire naturel

-> la valeur du nombre décimal est convertie en base 2

Valeur décimale entière positive	Représentation en binaire sur 8 bits
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
•••	
128	1000 0000
255	1111 1111



Représentation des entiers en binaire naturel

Nombre de Bits nécessaires pour représenter un nombre entier positif en binaire :

n =
$$[\log_2(x)] + 1 = \left[\frac{\ln(x)}{\ln(2)}\right] + 1 = [\ln(x)] + 1$$

Par exemple, pour représenter la valeur 65535, $\left\lceil \frac{\ln(65535)}{\ln(2)} \right\rceil + 1 = 16$ bits

nb: Les crochets représentent la partie entière.



Représentation des entiers en binaire naturel

-> le nombre de bits utilisés pour la représentation des nombres détermine la dynamique, c'est-à-dire la plage des valeurs que l'on peut coder

Nombre de bits utilisés	Valeur minimum	Valeur maximum	Nombre de valeurs	Type en langage C (peut varier)
8	0	255	256	unsigned char
16	0	65535	65536	unsigned short
32	0	4 294 967 295	4 294 967 296	unsigned int ou long
64	0	Oxffffffffff		unsigned long long



Addition des entiers en binaire naturel

-> Comme en base 10

A	+ B	= Retenue	Résultat
0	0	-	0
0	1	-	1
1	1	1	0



Multiplication des entiers en binaire naturel

-> Comme en base 10

A	х В	Résultat
0	0	0
0	1	0
1	1	1

$$-> 7 \times 3$$



Multiplication des entiers en binaire naturel

Remarque:

En binaire,

-> Multiplier par 2 revient à décaler tous les bits vers la gauche

(opérateur << du C)

-> Diviser par 2 revient à décaler tous les bits vers la droite

(opérateur >> du C)



En "complément à un"

- -> Le bit de poids fort positionné à 1 indique que le nombre est négatif
- -> la représentation en binaire s'obtient en calculant le **complément** de la valeur positive :
 - +7 se code 0000 0111
 - -7 se code 1111 1000
- -> Inconvénient: on a deux représentations pour le zéro (+0 et -0) soit (00000000 et 11111111)
- -> Le bit de gauche est utilisé pour représenter le signe



En "complément à deux"

- -> Le bit de poids fort positionné à 1 indique que le nombre est négatif
- -> la représentation en binaire s'obtient en calculant le complément à un de la valeur positive, puis en ajoutant 1
- -> la représentation en **complément à deux** permet de faire des additions d'entiers relatifs directement, et donc des soustractions



En "complément à deux" sur trois bits

valeur décimale	représentation binaire	valeur décimale en complément à 2	détail
0	000	0	0 + 0 + 0
1	001	1	2 ⁰ x1
2	010	2	2 ² x1
3	011	3	$2^{1}x1 + 2^{0}x1$
4	100	-4	$-2^2x1 + 0 + 0$
5	101	-3	$-2^2x1 + 0 + 2^0x1$
6	110	-2	$-2^2x1 + 2^1x1 + 0$
7	111	-1	$-2^2x1 + 2^1x1 + 2^0x1$



Obtention du "complément à deux" sur trois bits

- On calcule le complément à un
- Puis, on ajoute la valeur binaire 1
- On ne tient pas compte de la retenue à gauche
- Le résultat est faux s'il y a un dépassement de capacité (addition)

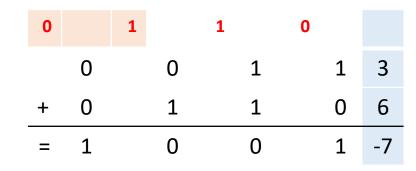
valeur décimale positive	représentation binaire	complément à 1	+1	valeur décimale du nb en complément à 2
0	000	111	1 000	0
1	001	110	111	-1
2	010	101	110	-2
3	011	100	101	-3
4	100	011	100	-4



"complément à deux", détection du dépassement de capacité :

Exemple avec des valeurs codées sur 4 bits : (comprises entre -8 et 7 donc) -> 2 nombres positifs

0		0	0	0	
	0	0	1	1	3
+	0	1	0	0	4
=	0	1	1	1	7







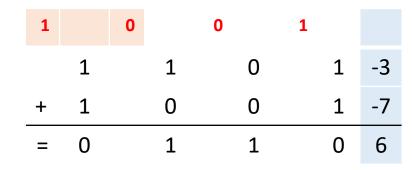


"complément à deux", détection du dépassement de capacité :

Exemple avec des valeurs codées sur 4 bits : (comprises entre -8 et 7 donc)

-> 2 nombres négatifs

1		1	0	0	
	1	1	0	1	-3
+	1	1	0	0	-4
=	1	0	0	1	-7









"complément à deux", détection du dépassement de capacité :

Exemple avec des valeurs codées sur 4 bits : (comprises entre -8 et 7 donc)

-> 1 nombres positif et 1 nombre négatif : le résultat est correcte si on ne tient pas compte de la retenue à gauche

0		0	0	1	
	1	1	0	1	-3
+	0	0	0	1	+1
=	1	1	1	0	-2

1		1	0	1	
	1	1	0	1	-3
+	0	1	0	1	+5
=	0	0	1	0	+2







"complément à deux", détection du dépassement de capacité :

2 cas -> les retenues sur les 2 rangs de gauche sont soit

$$-0-1$$

soit



Nombre de bits utilisés	Valeur minimum	Valeur maximum	Nombre de valeurs	Type en langage C
8	-128	+127	256	char
16	-32768	+32767	65536	short
32	-2147483648	+2147483647	4 294 967 296	int ou long
64	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	9223372036854775808 x 2	long long



1 – Combien faut-il de bits pour représenter les valeurs entières suivantes

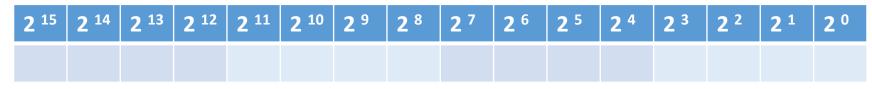
Valeur décimale	Nombre minimal de bits pour la représentation binaire	Type à choisir de préférence, en C
8		
13		
15		
125		
32 000		
66 700		
100 000 000 000		



2 – Donner la puissance de 2 immédiatement inférieure ou égale aux valeurs suivantes

Valeur décimale	puissance de 2 immédiatement inférieure ou égale
32 921	
153	
25	
9	
1	

En déduire la représentation binaire sur 16 bits de la valeur 32921 :

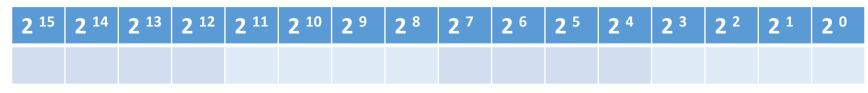


32921 =



3 – Répéter la division euclidienne par 2 de la valeur 32921 jusqu'à ce que ça ne soit plus possible

En déduire la représentation binaire sur 16 bits de la valeur 32921 :



32921 =



4 – Donner la représentation binaire en "complément à un" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à un
10 ₁₀	
0120 ₈	
101000002	
10 ₁₆	
FF ₁₆	



5 – Donner la représentation binaire en "complément à deux" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à deux
-10 ₁₀	
-128 ₁₀	
-1 ₁₀	
-1 ₁₀ -63 ₁₀	
-127 ₁₀	



6 – A quelles opérations arithmétiques correspondent les expressions en C suivantes:

Valeur	Opération réalisée	Valeur binaire sur 8 bits
10 ₁₀ >> 1		
128 ₁₀ >> 2		
1 ₁₀ << 1		
63 ₁₀ << 2		
128 ₁₀ << 1		



7 – Réaliser les opérations suivantes en binaire sur 8 bits (poser l'opération en binaire)

Opération	Valeur binaire du résultat sur 8 bits
50 / 2	
25 * 4	
80 - 80	
-128 + 127	
-30 + 5	
127 - 128	



Exercices: réponses

1 – Combien faut-il de bits pour représenter les valeurs entières suivantes

Valeur décimale	Nombre minimal de bits pour la représentation binaire	Type à choisir de préférence, en C
8	4	unsigned char
13	4	unsigned char
15	4	unsigned char
125	7	unsigned char
32 000	15	unsigned short
66 700	17	unsigned int
100 000 000 000	37	unsigned long long



Exercices: réponses

2 – Donner la puissance de 2 immédiatement inférieures ou égale au valeurs suivantes

Valeur décimale	puissance de 2 immédiatement inférieures ou égale
32 921	2 ¹⁵ (= 32768)
153	2 ⁷ (= 128)
25	2 4 (= 16)
9	$2^{3}(=8)$
1	2 0 (= 1)

En déduire la représentation binaire sur 16 bits de la valeur 32921 :

2 ¹⁵	2 14	2 ¹³	2 12	2 11	2 10	2 ⁹	28	2 ⁷	2 ⁶	2 ⁵	2 4	2 ³	2 ²	2 1	2 ⁰
1	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1

$$32921 = 2^{15} + 2^{7} + 2^{4} + 2^{3} + 2^{0}$$



Exercices : réponses

4 – Donner la représentation binaire en "complément à un" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à un
10 ₁₀	1111 0101
0120 ₈	1010 1111
101000002	0101 1111
10 ₁₆	1110 1111
FF ₁₆	0000 0000



Exercices: réponses

5 – Donner la représentation binaire en "complément à deux" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à deux
-10 ₁₀	1111 0110
-128 ₁₀	1000 0000
-1 ₁₀	1111 1111
-63 ₁₀	1100 0001
-127 ₁₀	1000 0001



Exercices: réponses

6 – A quelles opérations arithmétiques correspondent les expressions en C suivantes:

Valeur	Opération réalisée	Valeur binaire sur 8 bits
10 ₁₀ >> 1	division entière par 2	0000 0101
128 ₁₀ >> 2	division entière par 4	0010 0000
1 ₁₀ << 1	multiplication par 2	0000 0010
63 ₁₀ << 2	multiplication par 4	1111 1100
128 ₁₀ << 1	dépassement	0000 0000