1

Systèmes de numération

1-1 OBJECTIFS

1. Savoir définir la base d'un système de numération.
2. Savoir définir le rang, ou poids, d'un chiffre.
3. Savoir représenter un nombre de base b quelconque sous forme polynomiale.
4. Savoir déterminer la valeur décimale d'un nombre de base b quelconque.
5. Savoir convertir un nombre de base décimale en un nombre de base b quelconque (selon les deux procédés décrits).
6. Savoir convertir un nombre binaire en un nombre octal ou en un nombre hexadécimal et vice versa.
7. Savoir effectuer les quatre opérations ($+$, $-$, $\times$, $\div$) directement dans le système binaire naturel.
8. Savoir complémenter à 1 et à 2 un nombre binaire et savoir appliquer cette représentation à la soustraction.
9. Savoir écrire un nombre binaire sous forme normalisée.
10. Savoir comment un calculateur additionne et soustrait automatiquement.
11. Savoir coder un nombre décimal et un nombre binaire en Gray et vice versa.
12. Savoir coder un nombre décimal en BCD et vice versa.
13. Savoir coder un nombre décimal ou BCD en code «plus trois» et vice versa.
14. Connaître les principaux codes détecteurs et correcteurs d'erreurs et savoir comment la machine détecte et corrige une erreur.
15. Savoir coder et décoder une carte perforée selon le code Hollerith.
16. Savoir coder et décoder un ruban perforé selon le code ASCII.
17. Savoir ce qu'on entend par impulsions série et impulsions parallèle, leur avantage et leur inconvénient respectifs et savoir déterminer le nombre requis de fils pour émettre, dans ces deux modes, un nombre binaire donné.

1-2 INTRODUCTION

Ce chapitre expose la transition entre le codage et le fonctionnement intrinsèque d'un calculateur électronique automatique ou ordinateur.

Nous donnerons des exemples de codage pour le calcul numérique et nous vérifierons que l'algorithme d'une opération ne dépend pas du système de numération choisi.

Nous n'exposerons pas la théorie des quatre opérations élémentaires (addition, soustraction, multiplication, division) mais nous établirons un parallèle entre les techniques de ces opérations dans les systèmes de numération binaire et décimal.

Nous verrons de plus les premières notions de codage chiffré. Notions essentielles, car dans un ordinateur, même les caractères de l'alphabet (a, b, c, . . . , z) sont codés sous forme numérique binaire. Il faut donc se familiariser avec le calcul binaire et le codage.

L'ordinateur ne connaît que les chiffres, mais on peut, par son intermédiaire, commander un téléviseur couleur, écrire une partition, tracer des schémas ou des esquisses, etc.

1-3 BASE D'UN SYSTÈME DE NUMÉRATION

1-3-1 FORME POLYNOMIALE

On peut décomposer tout nombre N en fonction des puissances entières de la *base*[*] de son *système de numération*. Considérons, par exemple, le nombre *décimal* (base 10) 93452. On notera en indice la base du système de numération dans lequel le nombre N envisagé est écrit (sauf cas particuliers, on négligera de préciser les bases 2 et 10). On aura:

$$(93452)_{10} \equiv 9 \times 10^4 - 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$$

Par définition, à notre échelle, tout nombre élevé à la puissance 0 égale 1: $a^0 = 1$ quel que soit a.

Dans le système décimal nous disposons des dix symboles (appelés *chiffres*) notés 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Dans un nombre quelconque le chiffre de droite (2 dans l'exemple ci dessus) s'appelle le chiffre de *poids faible*, celui de gauche (9 dans notre exemple) s'appelle le chiffre de *poids fort*.

Exemple 1 Écrire $N = (27674)_{10}$ sous forme polynomiale et déterminer les chiffres de poids fort et faible.

Solution Nous aurons:

$$(27674)_{10} \equiv 2 \times 10^4 + 7 \times 10^3 + 6 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

chiffre de poids fort: 2, chiffre de poids faible: 4.

[*] La base d'un système de numération est le nombre de chiffres différents qu'utilise ce système de numération

On peut généraliser cette notion et écrire sous une forme plus abstraite tout nombre décimal N de n + 1 chiffres.

On aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 10^i$$

où a_i est un chiffre tel que $0 \leqslant a_i \leqslant 9$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 10 du chiffre de poids fort.

Le signe $\sum$, *sigma*, est l'opérateur sommation de tous les monômes à sa droite pour i variant de 0 à n. Le signe $\equiv$ signifie *identique à*.

1-3-2 RANG D'UN CHIFFRE

Le *rang* d'un chiffre d'un nombre de base b quelconque est égal à l'exposant de la base associée à ce chiffre dans la représentation polynomiale du nombre considéré. Soit par exemple, $N = (87672)_{10}$, alors: 2 est de rang 0, 6 est de rang 2, 8 est de rang 4.

Le rang des chiffres croît de la droite vers la gauche. Le rang du chiffre de droite est nul par définition.

Exemple 1 Soit $N = (23456)_{10}$. Déterminer:

a) le rang du chiffre 5 **(rép. 1)**

b) le rang du chiffre 3 **(rép. 3)**

REMARQUE: Au lieu de rang on peut employer «poids».

1-3-3 REPRÉSENTATION POLYNOMIALE D'UN NOMBRE N DE BASE b QUELCONQUE

Nous avons vu à la section 1-3-1 la décomposition de tout nombre décimal N en fonction de ses puissances entières de 10, soit:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 10^i$$

où a_i est un chiffre tel que $0 \leqslant a_i \leqslant 9$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 10 du chiffre de poids fort.

Il importe de noter que les symboles notés de 0 à 9 forment un ensemble ordonné. On aurait dû écrire en toute rigueur mathématique:

$a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ qui se lit:

a_i appartient ($\in$) à l'ensemble ordonné $\{0, 1, 2, \ldots, 9\}$ ou encore a_i est un élément de l'ensemble ordonné $\{0, 1, 2, \ldots, 9\}$.

Dans le cas d'une base b quelconque on respectera la même notation et les mêmes conventions. Tout nombre N de base b sera donc décomposable en fonction des puissances entières de b.

On aura donc:

$$N \equiv \sum_{i=0}^{i=n} a_i \times b^i$$

où $a_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers ≥ 0 et n est l'exposant de b du chiffre de poids fort.

Exemple 1 Dans le système à base 6, on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 6^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5\}$, les i sont des entiers ≥ 0 et n est l'exposant de 6 du chiffre de poids fort.

soit,

$$(54321)_6 \equiv 5 \times 6^4 + 4 \times 6^3 + 3 \times 6^2 + 2 \times 6^1 + 1 \times 6^0$$

Exemple 2 Dans le système à base 9, on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 9^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, les i sont des entiers ≥ 0 et n est l'exposant de 9 du chiffre de poids fort.

soit,

$$(87836)_9 \equiv 8 \times 9^4 + 7 \times 9^3 + 8 \times 9^2 + 3 \times 9^1 + 6 \times 9^0$$

Exemple 3 Comment écrire un nombre de base 12?

Solution

$$N \equiv \sum_{i=0}^{i=n} a_i \times 12^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots\}$, les i sont des entiers ≥ 0 et n est l'exposant de 12 du chiffre de poids fort.

Il nous manque deux chiffres. Selon la convention nord-américaine, nous prendrons les premières lettres A, B de l'alphabet.

On aura donc dans ce cas l'ensemble des chiffres suivants:

$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B\}$

Ce système de numération est dit *duodécimal*.

soit,

$$(9A73B)_{12} \equiv 9 \times 12^4 + A \times 12^3 + 7 \times 12^2 + 3 \times 12^1 + B \times 12^0$$

Exemple 4 Si b = 2, le système de numération est appelé *binaire* et l'on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 2^i$$

où $a_i \in \{0,1\}$, les i sont des entiers ≥ 0 et n est l'exposant de 2 du chiffre de poids fort.

soit,

$$(101101)_2 \equiv 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

1-3-4 VALEUR DÉCIMALE D'UN NOMBRE N DE BASE b QUELCONQUE

La *valeur décimale* d'un nombre N de base b s'obtient par sa forme polynomiale vue au paragraphe précédent.

Exemple 1 Déterminer la valeur décimale de N = (101101).

Solution Nous aurons:

$$\begin{aligned}(101101)_2 &\equiv 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &\equiv 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\ &\equiv 32 + 8 + 4 + 1 \\ &= (45)_{10}\end{aligned}$$

Exemple 2 Déterminer la valeur décimale du nombre *octal* (base 8) $N = (6734)_8$.

Solution Nous aurons:

$$\begin{aligned}(6734)_8 &\equiv 6 \times 8^3 + 7 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 \\ &\equiv 6 \times 512 + 7 \times 64 + 3 \times 8 + 4 \times 1 \\ &\equiv 3072 + 448 + 24 + 4 \\ &= (3458)_{10}\end{aligned}$$

Exemple 3 Déterminer la valeur décimale du nombre *hexadécimal* (base 16) $(A732)_{16}$.

Solution Nous aurons:

$$(A732)_{16} \equiv A \times 16^3 + 7 \times 16^2 + 3 \times 16^1 + 2 \times 16^0$$
$$\equiv 10 \times 4096 + 7 \times 256 + 3 \times 16 + 2$$
$$\equiv 40960 + 1792 + 48 + 2$$
$$= (42802)_{10}$$

1-4 CHANGEMENT DE BASE

1-4-1 PREMIER PROCÉDÉ DE CONVERSION D'UN NOMBRE DE BASE DÉCIMALE EN UN NOMBRE DE BASE b QUELCONQUE

Exemple 1 Convertir le nombre $N = (39487)_{10}$ en nombre octal.

Solution Nous avons vu à la section 1-3-3 que le nombre N cherché s'écrit:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 8^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 8 du chiffre de poids fort.

Le problème revient à déterminer les valeurs de a_i. Pour cela, appliquer l'*algorithme* suivant: chercher la plus grande puissance entière de 8 contenue dans $N = (39487)_{10}$, retrancher cette quantité de $(39487)_{10}$, considérer maintenant le reste obtenu et recommencer ce processus.

Il faut donc connaître les différentes puissances entières de 8. Elles figurent dans la table ci-dessous.

i	8^i
0	1
1	8
2	64
3	512
4	4096
5	32768

On aura successivement:

39487
− 32768 ----------→ 8^5

06719
− 4096 ----------→ 8^4

2623
− 512 ----------→ 8^3

2111
− 512 ----------→ 8^3

1599
− 512 ----------→ 8^3

1087
− 512 ----------→ 8^3

575
− 512 ----------→ 8^3

063
$7 \times 8 =$ 56 ----------→ 7×8^1

07 ----------→ 7×8^0

Donc:

$$N = (39487)_{10} \equiv 1 \times 8^5 + 1 \times 8^4 + 5 \times 8^3 + 7 \times 8^1 + 7 \times 8^0$$

On voit que le terme 8^2 est absent, d'où $a_2 = 0$, d'après notre relation.

On a donc:

$$N = (39487)_{10} = (115077)_8$$

Exemple 2 Convertir $N = (47375)_{10}$ en binaire.

Solution Comme précédemment pour 8, dressons une table des puissances entières de 2 et retranchons chaque fois la plus grande puissance entière de 2 possible.

i	2^i
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768

47375
− 32768 ----------→ 2^{15}
14607
− 8192 ----------→ 2^{13}
6415
− 4096 ----------→ 2^{12}
2319
− 2048 ----------→ 2^{11}
0271
− 256 ----------→ 2^{8}
015
− 8 ----------→ 2^{3}
7
− 4 ----------→ 2^{2}
3
− 2 ----------→ 2^{1}
1
− 1 ----------→ 2^{0}
0

Et l'on a:

$$N = (47375)_{10} \equiv 2^{15} + 2^{13} + 2^{12} + 2^{11} + 2^{8} + 2^{3} + 2^{2} + 2^{1} + 2^{0}$$

On voit que les termes $2^{14}, 2^{10}, 2^{9}, 2^{7}, 2^{6}, 2^{5}, 2^{4}$ sont absents. Nous en concluons donc que $a_{14} = a_{10} = a_9 = a_7 = a_6 = a_5 = a_4 = 0$.

Donc, $N = (47375)_{10} = (1011100100001111)_2$

RÉSUMÉ Dans ce premier algorithme on applique directement la formule:

$$N \equiv \sum_{i=0}^{i=n} a_i\, b^i$$

où $a_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers ≥ 0 et n est l'exposant de b du chiffre de poids fort.

On dresse une table donnant les différentes puissances entières de la base b du système de numération dans lequel on veut convertir le nombre décimal.

Au nombre décimal donné, on retranche la plus grande puissance entière de b possible. Cette puissance définit le rang du chiffre dans la

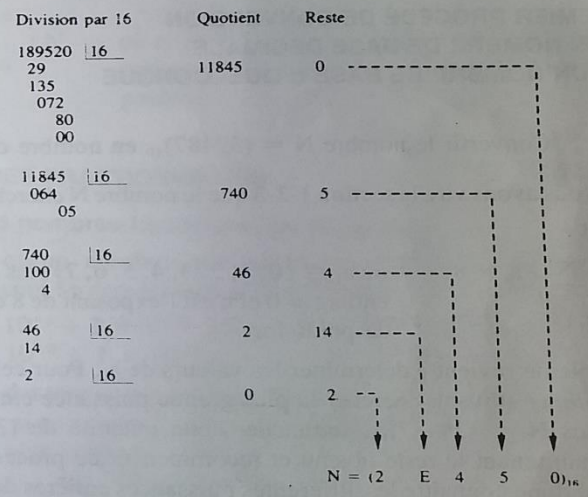
représentation en base b du nombre. Le nombre de fois (< b) qu'on retranche cette puissance définit le chiffre de ce rang. (En binaire, on ne peut avoir que 1 ou 0, la valeur du chiffre à écrire est donc immédiate).

1-4-2 DEUXIÈME PROCÉDÉ DE CONVERSION D'UN NOMBRE DE BASE DÉCIMALE EN UN NOMBRE DE BASE b QUELCONQUE

Cette méthode est simple et plus rapide que la précédente. Il est donc conseillé de l'utiliser dans tous les problèmes de conversion. Nous l'illustrerons à l'aide de deux exemples.

Exemple 1 Convertir le nombre $N = (189520)_{10}$ en hexadécimal.

Solution



Donc, $N = (189520)_{10} = (2E450)_{16}$

Vérification:

$$\begin{aligned} N &\equiv 2 \times 16^4 + 14 \times 16^3 + 4 \times 16^2 + 5 \times 16^1 + 0 \times 16^0 \\ &\equiv 2 \times 65536 + 14 \times 4096 + 4 \times 256 + 5 \times 16 + 0 \\ &\equiv 131072 + 57344 + 1024 + 80 \\ &= (189520)_{10} \end{aligned}$$

Exemple 2 Convertir le nombre $N = (231)_{10}$ en binaire.

Solution

Division par 2	Quotient	Reste
231 | 2		
115 | 2	115	1
57 | 2	57	1
28 | 2	28	1
14 | 2	14	0
7 | 2	7	0
3 | 2	3	1
1 | 2	1	1
	0	1

$N = (1\ 1\ 1\ 0\ 0\ 1\ 1\ 1)_2$

Donc, $N = (231)_{10} = (11100111)_2$

Vérification:

$$N \equiv 2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 2^0$$
$$\equiv 128 + 64 + 32 + 4 + 2 + 1$$
$$= (231)_{10}$$

RÉSUMÉ Cet algorithme consiste à diviser le nombre à convertir par la base du nouveau système et à conserver le reste. On répète ce processus en considérant chaque fois le quotient obtenu. On écrit ensuite tous les restes à partir de la fin et de gauche à droite, en les convertissant en lettres s'il y a lieu.

JUSTIFICATION DE LA MÉTHODE Soit N le nombre à convertir dans le système de numération de base b.

$N = q_0 b + r_0$ $q_0, q_1, q_2, \ldots q_{n-1}$: quotients
$r_0, r_1, r_2, \ldots r_n$: restes

$q_0 = q_1 b + r_1$
$q_1 = q_2 b + r_2$
⋮
$q_{n-3} = q_{n-2} b + r_{n-2}$
$q_{n-2} = q_{n-1} b + r_{n-1}$
$q_{n-1} = ob + r_n$

D'où:

$q_{n-2} = r_n b + r_{n-1}$
$q_{n-3} = (r_n b + r_{n-1})\, b + r_{n-2}$
$= r_n b^2 + r_{n-1} b + r_{n-2}$
⋮
$q_0 = r_n b^{n-1} + r_{n-1} b^{n-2} + \ldots + r_1$

D'où:

$$N = r_n b^n + r_{n-1} n^{n-1} + \ldots + r_1 b + r_0 b^0$$

Les restes r_i sont bien les coefficients de la décomposition polynomiale, donc

$$N \equiv \sum_{i=0}^{i=n} r_i b^i$$

où $r_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de b du chiffre de poids fort.

1-4-3 NOMBRES FRACTIONNAIRES

Rappel sur les nombres fractionnaires de base 10

Nous savons qu'un nombre fractionnaire décimal N, par exemple $(0{,}8237421)_{10}$ peut se décomposer sous la forme:

$$N \equiv 8 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3} + 7 \times 10^{-4} + 4 \times 10^{-5} + 2 \times 10^{-6} + 1 \times 10^{-7}$$

Les rangs des termes seront: $-1, -2, -3, -4, -5, -6$ et -7. D'une façon générale, si un nombre fractionnaire décimal r/s est inférieur à 1, on peut le mettre sous la forme:

$$r/s \equiv \sum_{i=1}^{i=n} a_i 10^{-i}$$

où $a_i \in \{0, 1, 2, \ldots, 9\}$, les i sont des entiers > 0 et $-n$ est l'exposant de 10 du chiffre de poids faible.

Si le système envisagé est de base b, on adoptera les mêmes conventions; on écrira donc un nombre fractionnaire sous la forme:

$$r/s \equiv \sum_{i=1}^{i=n} a_i b^{-i}$$

où $a_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers >0 et $-n$ est l'exposant de b du chiffre de poids faible.

Problème: Convertir un nombre fractionnaire inférieur à 1 de base b en décimal.

Exemple 1 Convertir $N = (0,1011001101)_2$ en décimal.

Solution Nous aurons:

$$N \equiv 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 0 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8} + 0 \times 2^{-9} + 1 \times 2^{-10}$$

Pour calculer ce nombre, il nous faut les valeurs décimales des puissances négatives de 2. Elles apparaissent dans la table suivante.

i	2^{-i}
1	0,5
2	0,25
3	0,125
4	0,0625
5	0,03125
6	0,015625
7	0,0078125
8	0,00390625
9	0,001953125
10	0,0009765625

la somme

0,5
+ 0,125
+ 0,0625
+ 0,0078125
+ 0,00390625
+ 0,0009765625

donne $N = (0,7001953125)_{10}$

Exemple 2 Convertir $N = (0,163)_8$ en décimal.

Solution Nous aurons:

$$N \equiv 1 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3}$$

i	8^{-i}
1	0,125
2	0,15625
3	0,001953125

la somme

0,125
+ 0,093750
+ 0,005859375

donne $N = (0,224609375)_{10}$

Problème: Convertir un nombre décimal fractionnaire en un nombre de base b.

Exemple 1 Convertir $N = (0,72145)_{10}$ en binaire.

Solution L'algorithme utilisé est analogue à celui de la deuxième méthode de conversion vue à la section 1-4-2, mais au lieu d'une division on aura une multiplication. Sa justification, de même nature, est laissée au lecteur. On aura donc successivement:

$0,72145 \times 2 = \boxed{1}, 44290$
$0,44290 \times 2 = \boxed{0}, 88580$
$0,88580 \times 2 = \boxed{1}, 77160$
$0,77160 \times 2 = \boxed{1}, 54320$
$0,54320 \times 2 = \boxed{1}, 08640$
$0,08640 \times 2 = \boxed{0}, 17280$
$0,17280 \times 2 = \boxed{0}, 34560$
$0,34560 \times 2 = \boxed{0}, 69120$
$0,69120 \times 2 = \boxed{1}, 38240$

Donc, $N = (0,72145)_{10} = (0,101110001)_2$

Il suffit donc d'écrire de gauche à droite les nombres encadrés pris de haut en bas.

Exemple 2 Convertir $N = (0{,}732)_{10}$ en octal.

Solution Nous aurons successivement:

$0{,}732 \times 8 = \boxed{5} \,.\, 856$

$0{,}856 \times 8 = \boxed{6} \,.\, 848$

$0{,}848 \times 8 = \boxed{6} \,.\, 784$

$0{,}784 \times 8 = \boxed{6} \,.\, 272$

$0{,}272 \times 8 = \boxed{2} \,.\, 176$

Donc $N = (0{,}732)_{10} = (0{,}56662)_8$

Vérification par l'expression polynomiale:

$$(0{,}56662)_8 \equiv 5 \times 8^{-1} + 6 \times 8^{-2} + 6 \times 8^{-3} + 6 \times 8^{-4} + 2 \times 8^{-5}$$
$$= 0{,}625 + 0{,}093750 + 0{,}011718750 \text{ en ne prenant que les trois premiers termes}$$
$$= (0{,}730468750)_{10}$$

Ce qui est une bonne approximation.

1-4-4 CONVERSION BINAIRE-OCTAL ET VICE VERSA

Quatre chiffres binaires ou *bits* permettent $2^4 = 16$ combinaisons et donc d'écrire les seize entiers de 0 à 15 selon le tableau ci-contre.

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Ce *code* est nommé code *binaire naturel* et dans notre cas plus spécifique code 8421. Chacun de ces chiffres représente le poids d'un bit. Ce code sera très souvent utilisé ultérieurement en techniques numériques: il serait donc utile qu'il vous soit familier dès à présent.

10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Nous allons maintenant illustrer comment convertir rapidement un nombre de base 2 en un nombre de base 8.

RÈGLE À partir de la virgule, grouper les bits par blocs de trois en allant vers la gauche pour la partie entière et vers la droite pour la partie fractionnaire. Convertir ensuite ces blocs en octal selon le code 8421.

Cette propriété découle du fait que la base 8 du système octal est une puissance entière de 2, en effet $8 = 2^3$.

Exemple 1

$N = (\ \underline{110}\ \ \underline{111}\ \ \underline{011}\ ,\ \underline{001}\ \ \underline{101}\ )_2$

$= (\ 6\ \ 7\ \ 3\ ,\ 1\ \ 5\ )_8$

Problème inverse:

Exemple 2 Convertir $N = (567{,}315)_8$ en binaire.

Solution Écrire, par blocs de trois bits, la valeur binaire des chiffres du nombre octal. On obtient dans ce cas:

$N = (101\ \ 110\ \ 111\ ,\ 011\ \ 001\ \ 101)_2$

Exemple 3 Convertir $N = (79182)_{10}$ en binaire.

Solution Convertissons ce nombre en nombre octal.

Division par 8	Quotient	Reste
79182 | 8		
71	9897	6
78		
62		
6		
9897 | 8		
18	1237	1
29		
57		
1		

1237 |8			
43	154	5	
37			
5			
154 |8			
74	19	2	
2			
19 |8			
3	2	3	
2 |8			
	0	2	

Donc, $N = (79182)_{10}$
$= (232516)_8$
$= (010 \quad 011 \quad 010 \quad 101 \quad 001 \quad 110)_2$

C'est une méthode plus rapide que la conversion directe en binaire car le nombre de divisions par 2 est trois fois plus grand que celui des divisions par 8.

1-4-5 CONVERSION BINAIRE-HEXADÉCIMAL ET VICE VERSA

La base du système de numération hexadécimal est aussi une puissance entière de 2, en effet $16 = 2^4$.

On a donc les mêmes propriétés que pour le système octal, mais cette fois on groupe les bits par blocs de quatre.

Exemple 1

$N = (\boxed{1001} \boxed{1100} \boxed{1011} \boxed{1010} , \boxed{0111})_2$
$= (\quad 9 \quad C \quad B \quad A \quad , \quad 7 \quad)_{16}$

Exemple 2 Convertir $N = (11432)_{10}$ en binaire.

Solution Convertissons ce nombre en nombre hexadécimal.

Division par 16	Quotient	Reste
11432 |16		
023	714	8
72		
08		

714 |16			
074	44	10	
10			
44 |16			
12	2	12	
2 |16			
	0	2	

Donc $N = (11432)_{10}$
$= (2 \quad C \quad A \quad 8)_{16}$
$= (0010 \quad 1100 \quad 1010 \quad 1000)_2$

Exemple 3

$N = (A \quad 7 \quad 8 \quad , \quad B \quad 3 \quad 2)_{16}$
$= (1010 \quad 0111 \quad 1000 \quad , \quad 1011 \quad 0011 \quad 0010)_2$

1-5 OPÉRATIONS ARITHMÉTIQUES EN BINAIRE

1-5-1 L'ADDITION

REMARQUE: Multiplier un nombre décimal par $(10)_{10}$ revient à lui ajouter un 0. De la même façon, multiplier un nombre binaire par $(10)_2$ revient à lui ajouter un 0: $1 \times 10 = 10$; $101 \times 10 = 1010$.

Cela nous permet de faire une autre remarque: ajouter à un nombre son égal revient à le multiplier par $(10)_2$, donc à lui ajouter un 0: $1 + 1 = 10$; $101 + 101 = 1010$.

Cela provient du fait que:

$2^{n-1} + 2^{n-1} = 2(2^{n-1}) = 2^n$; ce qui se traduit, par exemple, en binaire par:

1000	+	1000	=	10000
(n − 1) zéros		(n − 1) zéros		n zéros

Dressons la table d'addition:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ et report de 1

L'algorithme de l'addition des nombres binaires est le même que celui de l'addition des nombres décimaux.

Exemple 1

```
Reports:  1  1  1  1
          1  1  0  1  0  1
      +   1  0  1  1  0  0
      ---------------------
      1  1  0  0  0  0  1
```

Exemple 2

```
Reports:     1  1  1  1  1  1  1  1  1  1
             1  0  1  1  1  0  0  1  1  1  1  1  0  0  1
         +   1  1  0  1  1  1  1  0  1  0  1  0  1  1  0
      ---------------------------------------------------
      1  1  0  0  1  1  0  0  0  1  0  0  1  1  1  1
```

Pour 1 + 1 écrire 0 et reporter 1. Pour 1 + 1 et 1 de report, ce qui revient à 1 + 1 + 1, écrire 1 et reporter 1, ce qui revient à 11.

Il faut faire de nombreux exercices pour se familiariser avec l'addition en binaire.

1-5-2 LA SOUSTRACTION

Lorsque le diminuteur est plus petit que le diminuende, on aura un résultat de signe positif. Dans le cas contraire, intervertir les termes et affecter le résultat du signe − (moins).

Exemple en décimal:

```
   632
 − 475
 -----
 + 157
```

Au lieu d'effectuer la soustraction 475 − 632, effectuer 632 − 475 = 157 et écrire comme résultat − 157.

Dressons la table de soustraction:

0 − 0 = 0
0 − 1 = 1 et retenue de 1
1 − 0 = 1
1 − 1 = 0

La retenue de 1 sera retranchée du chiffre de rang supérieur.

Exemple 1

```
  0    0 0
  X 0 X X 0 1 1
−   1 0 1 1 1 1
  -------------
    1 0 1 1 0 0
```

Lorsqu'on a une retenue, rayer le 1 de rang supérieur et le remplacer par 0.

Exemple 2

```
  0 1 1 0
  1 0 0 X 0 1 1
−   1 0 1 1 1 1
  -------------
    0 1 1 1 0 0
```

Dans ce cas, rayer pour la retenue le premier 1 rencontré, le remplacer par 0 et les 0 intermédiaires par des 1, car:

```
  1 0 0 0
−       1
  -------
    1 1 1
```

1-5-3 LA MULTIPLICATION

La disposition des nombres à multiplier est la même en binaire qu'en décimal, l'algorithme est aussi le même.

La table de multiplication est particulièrement simple:

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Si on multiplie par 1, écrire le multiplicande et si on multiplie par zéro, écrire 0.

Exemple 1

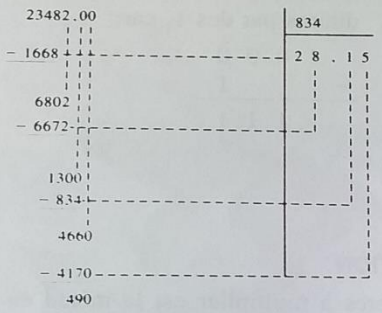
```
      1 0 1 1 0 1
  ×       1 0 1
  -----------------
      1 0 1 1 0 1
  1 0 1 1 0 1 .        <------  . décalage dû au zéro de 101
  -----------------
  1 1 1 0 0 0 0 1
```

Exemple 2

```
                  1 1 0 1 1 0 1
              × 1 0 1 0 0 1 1
  ---------------------------------
                  1 1 0 1 1 0 1
                1 1 0 1 1 0 1
              0 0 0 0 0 0 0 .
            1 1 0 1 1 0 1 . .
          1 1 0 1 1 0 1 .
  ---------------------------------
  1 0 0 0 1 1 0 1 0 1 0 1 1 1
```

1-5-4 LA DIVISION

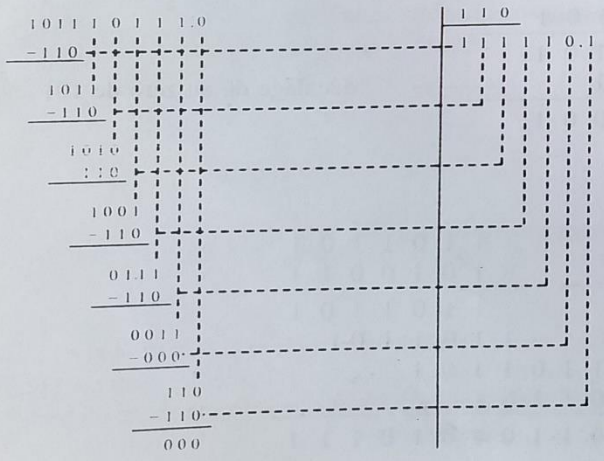
RAPPEL: soit à diviser $(23482)_{10}$ (dividende) par $(834)_{10}$ (diviseur). On pose la division:



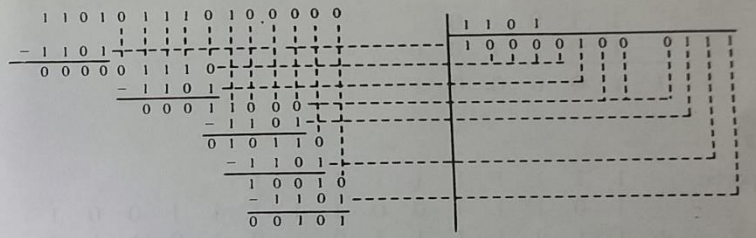
Cette disposition classique de la division est enseignée dans les classes élémentaires.

Pour la division en binaire, c'est la même disposition et le même algorithme. Les chiffres du quotient, plus simples, sont 0 ou 1; 1 lorsque le diviseur est plus petit que le dividende et 0 dans l'autre cas.

Exemple 1



Exemple 2



1-6 LA COMPLÉMENTATION

1-6-1 COMPLÉMENT À 1

En décimal, on forme le *complément à* 9 d'un nombre, par exemple 78543, en soustrayant de 9 chaque chiffre de ce nombre. Dans notre cas, on obtient 21456. La somme de ces nombres donne 99999.

En binaire, on forme le *complément à* 1 d'un nombre en soustrayant de 1 chaque bit de ce nombre. Le complément à 1 de 1 0 1 1 0 1 1 1 0 0 1 0, par exemple, est 0 1 0 0 1 0 0 0 1 1 0 1.

Donc, pour obtenir le complément à 1 d'un nombre binaire il suffit de complémenter chaque bit: lorsqu'on a 1, écrire 0 et lorsqu'on a 0, écrire 1. La somme d'un nombre binaire et de son complément à 1 est un nombre binaire uniquement composé de 1.

1-6-2 COMPLÉMENT À 2

En décimal, on forme le *complément à* 10 ou complément *vrai* d'un nombre, par exemple 673425, en soustrayant de 10 le chiffre de rang 0 et de 9 les autres. Dans notre cas on obtient 326575.

Effectuons la somme de ces deux nombres

$$\begin{array}{r} 673425 \\ +\ 326575 \\ \hline 1000000 \end{array}$$

on obtient:

Trouver le complément à 10 d'un nombre revient à le soustraire de la puissance de 10 immédiatement supérieure.

$$\begin{array}{r} 1000000 \\ -\ 673425 \\ \hline 326575 \end{array}$$

donne:

En binaire, trouver le *complément à* 2 d'un nombre revient à le soustraire de la puissance de 2 immédiatement supérieure. Par exemple le complément à 2 de

1 1 0 1 0 1 1 0 1 0 est le résultat de la soustraction

```
  1 0 0 0 0 0 0 0 0 0 0 0
- 1 1 0 1 0 1 1 0 1 0
-----------------------
soit,  0 0 1 0 1 0 0 1 1 0
```

Trouver le complément à 2 revient aussi à trouver le complément à 1 (en complémentant chaque bit) et à ajouter 1 au résultat. Dans le cas précédent on obtiendrait:

```
  0 0 1 0 1 0 0 1 0 1
+               1
-------------------
  0 0 1 0 1 0 0 1 1 0
```

Une troisième méthode consiste à conserver tous les bits à partir de la droite jusqu'au premier 1 compris et de changer les autres bits de 0 en 1 ou de 1 en 0 comme pour le complément à 1. Dans notre cas on aurait:

0 0 1 0 1 0 0 1 [1 0] bits conservés.

Exemple 1 Trouver le complément à 2 de 1 0 1 1 0 1 1 0 1 0 0 0

Réponse: 0 1 0 0 1 0 0 1 1 0 0 0

1-6-3 SOUSTRACTION PAR COMPLÉMENTATION À 1 ET ADDITION

Voyons maintenant une propriété du complément des nombres qui permettra de justifier un autre algorithme pour la soustraction. Nous tirerons cette propriété d'un exemple sur des nombres décimaux.

Soit la soustraction en décimal

```
  17382
- 12457
-------
+ 04925
```

résultat:

Si nous prenons le complément à 9 du diminuteur, nous obtenons 87542 qui ajouté au diminuende donne:

```
  1 7 3 8 2
+ 8 7 5 4 2
-----------
1 0 4 9 2 4
  └──→ + 1
-----------
  0 4 9 2 5
```

et en ajoutant le dernier 1 on obtient de la soustraction

le résultat:

Cette façon de procéder s'applique également aux nombres binaires en utilisant le complément à 1.

Exemple 1 Soit la soustraction suivante:

```
  1 1 0 1 0 1 1 1 0 1  --------→  1 1 0 1 0 1 1 1 0 1
- 1 0 1 1 1 0 0 1 1 1              + 0 1 0 0 0 1 1 0 0 0 : complément à 1
                                  1 0 0 0 1 1 1 0 1 0 1
                                  └──→ + 1
                                  -----------------
résultat:                         0 0 0 1 1 1 0 1 1 0
```

Exemple 2 Soit la soustraction suivante:

```
  1 0 1 0 0 1 1 1  --------→  1 0 1 0 0 1 1 1
- 1 0 0 1 1 0 0 1              + 0 1 1 0 0 1 1 0 : complément à 1
                               1 0 0 0 0 1 1 0 1
                               └──→ + 1
                               -----------------
résultat:                      0 0 0 0 1 1 1 0
```

On peut déjà prévoir que l'on peut se passer d'ajouter 1 à la fin si on prend le complément à 2 du nombre à soustraire. Dans ce cas, on aura ajouté 1 au moment de la complémentation et le résultat sera le même.

1-6-4 SOUSTRACTION PAR COMPLÉMENTATION À 2 ET ADDITION

Exemple 1 Soit la soustraction suivante:

```
  1 1 0 1 1 0 1 1 1  --------→  1 1 0 1 1 0 1 1 1
- 1 0 1 0 1 1 1 0 1              + 0 1 0 1 0 0 0 1 1 : complément à 2
                                1)0 0 1 0 1 1 0 1 0 : résultat
```

débordement à éliminer

Exemple 2

REMARQUE 1: 11010 est équivalent à 0 0 0 1 1 0 1 0, les zéros en tête du nombre n'étant pas significatifs.

REMARQUE 2: Dans cet exemple on a remplacé les chiffres manquants en tête du nombre par des zéros, ce qui nous a donné des 1 au moment de la complémentation. Cette remarque est très importante sinon on ne retrouve pas le résultat de la soustraction.

Soit la soustraction suivante:

```
 1 0 1 1 1 0 1 1 1 0 1        voir remarques 1 et 2
-    1 0 1 1 0 0 1 1 0   ---->  1 0 1 1 1 0 1 1 1 0 1
                              + 1 1 0 1 0 0 1 1 0 1 0
                              1) 1 0 0 0 1 1 1 0 1 1 1 : résultat
```

débordement à éliminer

Nous allons étudier maintenant une forme d'*emmagasinage* de nombres dans un ordinateur.

1-6-5 FORME NORMALISÉE

Un nombre binaire placé dans une mémoire de machine peut être fractionnaire. Pour des raisons pratiques de circuiterie et de construction on est obligé de placer ces nombres sous *forme normalisée*. Il est clair qu'il est impossible de prévoir tous les cas possibles de position de la *virgule arithmétique*. Une des formes normalisées utilisées dans les machines automatiques est la représentation à *virgule fixe*.

Dans ce cas, pour des raisons d'efficacité, les machines sont construites pour représenter les nombres sous la forme:

$+ \; 0 \; \Lambda \; \ldots\ldots$

$- \; 0 \; \Lambda \; \ldots\ldots$ où Λ représente la virgule fixe différente parfois de la virgule arithmétique classique.

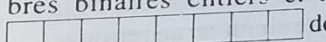
En décimal, par exemple, on peut normaliser les nombres de la façon suivante, à huit *caractères*

Nombres	Nombres normalisés à huit caractères
1282	+ Λ 1282000
31	+ Λ 0031000
7,4	+ Λ 0007400
0,032	+ Λ 0000032
− 1,32	− Λ 0001320

Le signe Λ n'est pas un caractère codé.

Le *facteur d'alignement* est le nombre de positions de caractères comprises entre la virgule fixe et la virgule arithmétique. Dans notre cas le facteur d'alignement est de 4.

Pour simplifier cet ouvrage nous ne considérerons que des nombres binaires entiers et nous choisirons un mot de huit caractères



dont une *case*, la première à gauche, sera réservée au signe. Le facteur d'alignement dans notre cas sera de 7.

Exemple 1 Représenter $(27)_{10}$ en binaire dans notre cas de normalisation.

Solution

$(27)_{10} = (11011)_2$ d'où:

+	0	0	1	1	0	1	1

Exemple 2 Même problème pour $(-37)_{10}$.

Solution

$(-37)_{10} = -(100101)_2$ d'où:

−	0	1	0	0	1	0	1

Exemple 3 Déterminer le plus grand nombre que l'on peut représenter de cette façon. Ce sera + 1 1 1 1 1 1 1 qui correspond à 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127. Si nous voulons représenter un nombre plus grand que 127 nous aurons, dans notre cas, un débordement. Notre système ne comporte pas assez de cases pour placer un tel nombre.

Les gros ordinateurs ont jusqu'à 16 ou 32 bits pour les nombres entiers. Le miniordinateur PDP8 de Digital, par exemple, en a 12.

Les nombres fractionnaires ou les très grands nombres sont écrits sous forme d'une fraction normalisée et d'un exposant. On a alors une représentation à *virgule flottante*.

1-6-6 NOMBRES POSITIFS ET NÉGATIFS BINAIRES NORMALISÉS À HUIT CARACTÈRES

Les signes + et − ne sont pas assimilables tels quels par un ordinateur lequel ne connaît que deux états: 0 et 1. On convient donc de les représenter par un bit qui occupera la case de gauche du jeu de cases d'écriture du nombre considéré. Ce bit est appelé le *bit de signe*.

On peut, par exemple, représenter le signe + par 0 et le signe − par 1. C'est la convention généralement adoptée.

De plus, dans la plupart des cas, on représente les nombres négatifs sous la forme complément à 2.

C'est le mode d'écriture que nous retiendrons. Nous représenterons les nombres sous forme normalisée à huit caractères. Les nombres négatifs seront représentés sous leur forme complément à 2.

Exemple Représenter sous forme binaire normalisée à huit caractères les nombres décimaux.

Nombres	Solutions							
32	0	0	1	0	0	0	0	0
39	0	0	1	0	0	1	1	1
41	0	0	1	0	1	0	0	1
−27	1	1	1	0	0	1	0	1
−32	1	1	1	0	0	0	0	0
− 1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0

1-6-7 CALCUL AUTOMATIQUE

Grâce à cette façon de procéder, un calculateur additionne et soustrait automatiquement à l'aide d'un seul *circuit* dit *additionneur*. On obtiendra le résultat avec son signe. En voici quelques exemples.

Exemples Soit les additions suivantes, en représentations décimale à gauche, binaire normalisée à droite.

a)
```
  27          00011011
+ 61        + 00111101
------      ----------
  88        01011000 = (88)10
```

b)
```
  61          00111101
- 27        + 11100101
------      ----------
+ 34       1) 00100010 = (34)10
```
débordement à éliminer

c)
```
  27          00011011
- 61        + 11000011
------      ----------
- 34        11011110
```

Le résultat est de signe −, en prenant le complément à 2 du résultat on a 000100010 donc − 34. On obtient donc le résultat cherché avec le signe − et sous la forme complément à 2.

d)
```
  61          00111101
+ 88        + 01011000
------      ----------
 149        10010101 = (149)10
```


Dans ce cas, on a un changement de signe du résultat. L'ordinateur détectera ce changement de signe et avertira l'opérateur qu'il y a débordement. C'est notre cas puisque 149 > 127, capacité maximale de nos *registres*.

Nous venons de voir comment un calculateur automatique effectue les quatre opérations élémentaires, la multiplication et la division se ramenant à des additions.

1-7 CODES

1-7-1 CODE GRAY OU BINAIRE RÉFLÉCHI

Le code binaire que nous avons vu jusqu'à maintenant s'appelle le code binaire naturel. Il existe de nombreux autres codes dont le code *Gray*, aussi appelé code *binaire réfléchi*. Dans les conversions d'une *grandeur analogique* (par exemple, la position de l'axe d'un moteur) en une *grandeur numérique* on a besoin d'un code dans lequel les grandeurs successives ne diffèrent que d'un caractère. Cela évite des *erreurs de détection*. Soit, le passage de 7 à 8, par exemple. Selon la règle binaire naturelle, on passera de 0111 à 1000: les quatre bits changent. Dans les états intermédiaires, s'ils ne changent pas tous en même temps, on peut détecter des valeurs erronées.

Le tableau ci-dessous donne l'équivalent en code Gray des entiers de 0 à 15.

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

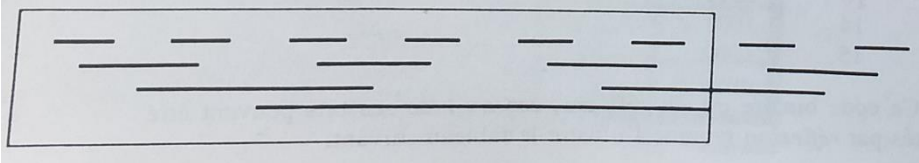
Ce code binaire est dit réfléchi, car n−1 de ses bits peuvent être générés par *réflexion* comme l'illustre le tableau suivant.

0 0	0 0 0	0 0 0 0
0 1	0 0 1	0 0 0 1
1 1	0 1 1	0 0 1 1
1 0	0 1 0	0 0 1 0
	1 1 0	0 1 1 0
	1 1 1	0 1 1 1
	1 0 1	0 1 0 1
	1 0 0	0 1 0 0
		1 1 0 0
		1 1 0 1
		1 1 1 1
		1 1 1 0
		1 0 1 0
		1 0 1 1
		1 0 0 1
		1 0 0 0

Ce tableau illustre une façon d'écrire un nombre décimal en Gray.

Une deuxième façon est donnée par la figure suivante qui est le schéma d'une règle codée en Gray. Les traits pleins peuvent correspondre à une fenêtre transparente et à des 1, l'absence de traits à une absence de fenêtres et à des 0. On détecte la lumière passant à travers ces fenêtres et on a immédiatement en code Gray la position du flux lumineux.

Le *codeur-décodeur* Gray de la figure suivante permettra, par exemple, de relever directement et sous forme numérique, la position longitudinale d'un chariot coulissant d'une machine-outil.



1-7-2 CODE BCD (Binary Coded Decimal) OU Décimal Codé Binaire (DCB)

Ce code conserve les avantages du système décimal et du code binaire. Il est utilisé par les machines à calculer.

On fait correspondre à chaque caractère du système décimal un mot du code binaire de quatre bits, on a alors:

code décimal	0	1	2	3	4	5	6	7	8	9
code BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

On remarque que si on supprime les zéros de gauche on retrouve le nombre binaire naturel.

Pour coder le nombre décimal 8 6 3 8 7, par exemple, écrire:

1000	0110	0011	1000	0111
8	6	3	8	7

Les opérations arithmétiques effectuées dans ce code sont plus compliquées qu'en binaire naturel. Par exemple:

9 2 3 7	1001	0010	0011	0111
+ 8 1	+ 0000	+ 0000	+ 1000	+ 0001
9 3 1 8	1001	0010	1011	1000
	9	2	?	8

Nous rencontrons ici un mot codé qui ne correspond pas à une valeur connue. Pour résoudre ce problème, on peut ajouter $(6)_{10} = (0\ 1\ 1\ 0)_2$ à ce mot codé inconnu, ce qui donnera

$$\begin{array}{r} 1011 \\ 0110 \\ \hline 10001 \end{array}$$

et ajouter 1 (report) au nombre suivant.

Dans ce cas, le résultat sera:

1001	0011	0001	1000
9	3	1	8

Cette difficulté provient du fait que quatre bits donnent $2^4 = 16$ états dont 10 seulement servent à coder les chiffres décimaux.

Dans les premiers ordinateurs on a cherché à contourner cette difficulté en utilisant le code «*plus trois*» étudié à la section 1-7-4.

1-7-3 CODE ASCII
(American Standard Code for Information Interchange)

Ce code est une norme presque universelle dans les *transmissions*. Il comprend sept ou huit caractères. Le huitième caractère dit de *parité* sert à détecter les erreurs de transmission. On étudiera ce mode de *détection d'erreurs* plus loin.

La table ci-dessous donne le code ASCII

b_4	b_3	b_2	b_1	b_7 b_6 b_5 →	000	001	010	011	100	101	110	111
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	TC_7 (DLE)	SP	0	@	P	`	p
0	0	0	1	1	TC_1 (SOH)	DC_1	!	1	A	Q	a	q
0	0	1	0	2	TC_2 (STX)	DC_2	"	2	B	R	b	r
0	0	1	1	3	TC_3 (ETX)	DC_3	#	3	C	S	c	s
0	1	0	0	4	TC_4 (EOT)	DC_4	¤	4	D	T	d	t
0	1	0	1	5	TC_5 (ENQ)	TC_8 (NAK)	%	5	E	U	e	u
0	1	1	0	6	TC_6 (ACK)	TC_9 (SYN)	&	6	F	V	f	v
0	1	1	1	7	BEL	TC_{10} (ETB)	'	7	G	W	g	w
1	0	0	0	8	FE_0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9	FE_1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	10	FE_2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	11	FE_3 (VT)	ESC	+	;	K	[	k	{
1	1	0	0	12	FE_4 (FF)	IS_4 (FS)	,	<	L	\	l	|
1	1	0	1	13	FE_5 (CR)	IS_3 (GS)	–	=	M	]	m	}
1	1	1	0	14	SO	IS_2 (RS)	.	>	N	^	n	‾
1	1	1	1	15	SI	IS_1 (US)	/	?	O	_	o	DEL

CCITT-11540

Exemples de codes Le code binaire est donné par $b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1$, ($b_1$ est le bit de poids faible).

Le code de:

U est 55_{16} soit 1010101 en binaire
F est 46_{16} soit 1000110
K est $4B_{16}$ soit 1001011

réciproquement, le code correspondant à:

43_{16} ou 1000011 est C
0110011 soit 33_{16} est 3

1-7-4 CODE «PLUS TROIS» *(Excess 3)*

L'inconvénient cité ci-dessus à propos des opérations arithmétiques en BCD a été contourné dans les ordinateurs de la première génération à l'aide du code «plus trois». Le code plus trois des nombres décimaux de 0 à 9 est le code BCD auquel on ajoute 3, comme son nom l'indique, ce qui donne:

0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

En général, pour avoir le résultat d'une addition en BCD + 3 il faut ajouter 3 si on a un report et retrancher 3 dans le cas contraire ($-3 = -0011 = 1101$ en complément à 2).

Exemple 1 Avec report

7	devint en BCD plus trois		1010
+ 5		+	1000
12		0001	0010
		0011	0011
		0100	0101 : BCD + 3

Exemple 2 Sans report

```
   5   devient en BCD plus trois  1000
 + 3                              0110
 ---                              ----
   8                              1110
                                + 1101
                                ------
                              1) 1011   : BCD + 3
```

débordement à éliminer ⟵

1-7-5 CODES DÉTECTEURS D'ERREURS

Dans un code de quatre bits par exemple (ou de sept bits pour le code ASCII) une erreur simple sur un bit peut faire apparaître un autre *mot de code*. Par exemple, en BCD le mot de code 0010 (2) transmis par erreur 0110 (6, inversion du 3e bit due aux parasites de la ligne) sera interprété comme un 6 par le *récepteur*, mais ce sera une *donnée* erronée.

Il existe plusieurs façons de détecter ce type d'erreurs. La plus utilisée est celle du *bit de parité paire* ou *impaire*.

Exemple de code de parité paire en BCD:

	8421	p
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

p: bit de parité paire
p = 0 si le nombre de 1 du mot de code est pair
= 1 autrement.

Si on a une erreur simple dans le mot de code ou dans le bit de parité, elle sera immédiatement détectée. Le récepteur pourra demander une retransmission jusqu'à réception du bon mot.

Il existe d'autres codes de détection d'erreurs. Citons, par exemple, le code «*deux de cinq*» qui représente les dix combinaisons possibles de deux 1 dans un mot de cinq bits, ce qui donne:

0	00011
1	11000
2	10100

3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101

1-7-6 CODES DÉTECTEURS ET CORRECTEURS D'ERREURS

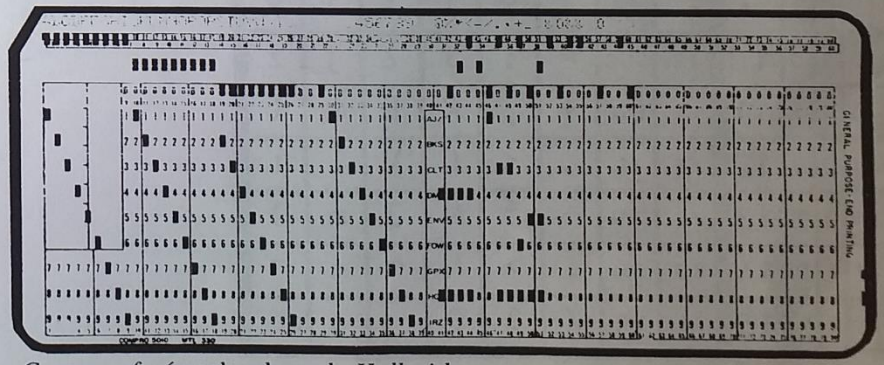
Le système *correcteur d'erreurs* suivant est utilisé sur les *bandes magnétiques*. On envoie une série de mots codés (verticalement sur le tableau) et en fin de *message* on envoie les parités paires *longitudinale* et *verticale*.

	3	4	5	7	parité longitudinale
	0	0	0	0	0
	0	1	1	1	1
	1	0	0	1	0
	1	0	1	1	1
parité verticale	0	1	0	1	

Si, par exemple, le mot de code de 5 comporte une erreur, disons que son 2e bit est erroné, la machine localisera une erreur dans la 3e colonne et dans la 2e ligne et la corrigera immédiatement.

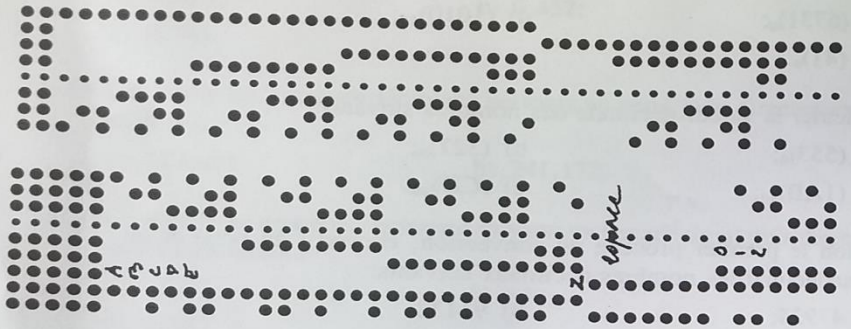
1-8 CODAGE

1-8-1 CARTE PERFORÉE SELON LE CODE HOLLERITH



Carte perforée selon le code *Hollerith*

1-8-2 RUBAN PERFORÉ SELON LE CODE ASCII



Disposition des perforations. Un trou représente un 1.

o o o o o ; o o o

perforation pour l'entraînement

bit de parité

Exemple Le code de la lettre «B» qui est 42_{16} sera représenté par

. o . . . . . o .

1-8-3 IMPULSIONS

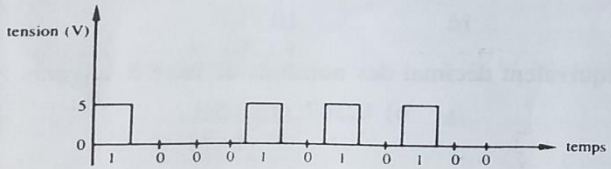
Pour représenter l'*état* 0 ou 1 d'une donnée à transmettre, les calculateurs utilisent deux *niveaux de tension*. Les plus répandus sont

0 volt pour l'état 0
5 volts pour l'état 1.

Cette *logique* est connue sous le nom de logique *positive* (l'inverse, 0 volt pour l'état 0 et −5 volts pour l'état 1 est appelé logique *négative*).

a) Impulsions série

Pour transmettre un nombre, on envoie donc dans le temps deux niveaux ou *impulsions* de tension selon, par exemple, la configuration *série* suivante:

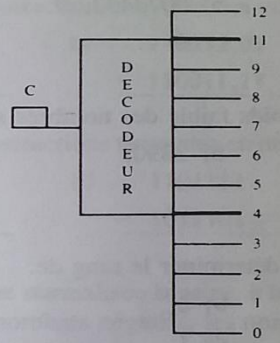


Ces tensions sont soit positives soit négatives. L'important est d'avoir deux niveaux de tension nettement distincts.

On n'a donc besoin dans ce cas que d'un fil de transmission d'où une économie de câblage. Cela présente un inconvénient: la transmission d'un mot s'étire dans le temps.

b) Impulsions parallèle

Lorsqu'on veut actionner les marteaux d'une perforatrice, par exemple, on actionne tous les marteaux nécessaires d'une même colonne en même temps. Pour cela on commande les électro-aimants par des impulsions *parallèle*. La figure ci-dessous illustre la perforation de C en parallèle.



Avantage d'une transmission par impulsions parallèle:
— Rapidité: on a tout le mot codé en même temps.

Inconvénient:
— Nécessite beaucoup de fils de transmission. En général plus coûteuse.

PROBLÈMES

Voir le corrigé des problèmes à l'appendice A. Si votre réponse diffère de celle y figurant, rechercher l'erreur commise.

1-1 Écrire les nombreux décimaux suivants sous forme polynomiale:

a) 27; b) 301;
c) 08641; d) 39472;
e) 473283.

1-2 Déterminer le chiffre de poids fort des nombres suivants:

a) 3812; b) 0073432;
c) 162622.

1-3 Déterminer le chiffre de poids faible des nombres suivants:

a) 7432; b) 3890;
c) 170003.

1-4 Soit le nombre 49736421, déterminer le rang de:

a) 9; b) 6;
c) 2; d) 1;
e) 7; f) 4.

1-5 Donner, sous forme d'un ensemble ordonné, les symboles que vous utiliseriez dans les systèmes de numération de base:

a) 3; b) 7;
c) 4; d) 11;
e) 5; f) 13;
g) 16.

1-6 Décomposer les nombres suivants en fonction des puissances de la base de leur système de numération:

a) $(101101110)_2$; b) $(B9107A)_{12}$;
c) $(EA735)_{16}$; d) $(654321)_7$;
e) $(6583)_8$.

1-7 Calculer la valeur décimale des nombres suivants:

a) $(CD1)_{16}$; b) $(101110101)_2$;
c) $(6731)_8$; d) $(1010)_{16}$;
e) $(41)_8$.

1-8 Calculer la valeur décimale des nombres suivants:

a) $(553)_6$; b) $(327)_8$;
c) $(1AB)_{12}$; d) $(CD)_{16}$.

1-9 Selon le premier procédé de conversion, convertir en octal et en hexadécimal les nombres décimaux suivants:

a) 47934; b) 91132;
c) 743; d) 121.

1-10 Selon le premier procédé de conversion, convertir en binaire les nombres décimaux suivants:

a) 283; b) 151;
c) 1142.

1-11 Selon le deuxième procédé de conversion, convertir en octal et en hexadécimal les nombres décimaux suivants;

a) 31432; b) 87947;
c) 653; d) 831.

1-12 Selon le deuxième procédé de conversion, convertir en binaire les nombres décimaux suivants:

a) 253; b) 432;
c) 512; d) 1024.

1-13 Déterminer l'équivalent décimal des nombres binaires suivants:

a) 11,011011; b) 1,101101;
c) 1,0001011.

1-14 Déterminer l'équivalent décimal des nombres de base 8 suivants:

a) 0,252; b) 0,347;
c) 0,151.

1-15 Convertir en binaire, en octal et en hexadécimal les nombres décimaux suivants:

a) 0,375; b) 0,432;

c) 0,741.

1-16 Convertir les nombres décimaux suivants en octal puis directement en binaire:

a) 123,431; b) 241,172.

1-17 Convertir les nombres décimaux suivants en hexadécimal puis directement en binaire:

a) 653; b) 831;

c) 87947.

1-18 Convertir en hexadécimal et en octal les nombres binaires suivants:

a) 110111,0110111101; b) 1110001111,010101;

c) 101110111011,0101111.

1-19 Effectuer les additions suivantes en binaire:

a) 1011,1101 + 110,1011

b) 110111,01 + 110011,11

c) 101111101 + 111111011

d) 1101111 + 1110000

1-20 Effectuer les soustractions suivantes en binaire:

a) 1011,1101 − 110,1011

b) 110111,01 − 110011,11

c) 111111011 − 101111101

d) 1110000 − 1101111

1-21 Effectuer les multiplications suivantes en binaire:

a) 11011,01 × 101,01

b) 10111,11 × 110

c) 11001,011 × 1110

d) 110111 × 10001

1-22 Effectuer les divisions suivantes en binaire:

a) 101101 | 1101 b) 1101111 | 1010

c) 11011101 | 1001 d) 11011 | 11

1-23 Trouver: A) le complément à 1; B) le complément à 2 des nombres binaires suivants:

a) 1101110111; b) 1011011101; c) 11011;

d) 10110111; c) 1000001; f) 1100111;

g) 110111; h) 10110101; i) 01001110.

1-24 Effectuer les soustractions suivantes en utilisant le complément à 1:

a) 1011,1101 − 110,1011

b) 110111,01 − 110011,11

c) 111111011 − 101111101

1-25 Effectuer les soustractions suivantes en utilisant le complément à 2:

a) 1110000 − 1101111

b) 1101111 − 1011101

c) 100111101 − 11011110

1-26 Écrire sous forme normalisée binaire, à huit caractères, en complémentant à 2 les nombres négatifs, les nombres décimaux suivants:

a) −17; b) 31;

c) −42; d) 51;

e) −128.

1-27 Inclure le bit de signe des nombres ci-dessus.

1-28 Donner sous forme normalisée le résultat des opérations suivantes en indiquant son équivalent décimal.

a) (31 − 17); b) (42 − 51);

c) (− 128 − 17); d) (31 − 42).

1-29 Écrire en code binaire réfléchi (Gray) les nombres suivants:

a) 27; b) 31;

c) 19; d) 23.