Baccalauréat STI2D, classe de première : Enseignement Transversal	
 **Chaîne d'information**	

Sciences et Technologies de l'Industrie et du Développement Durable

3 : La chaîne d'information (Codage : Binaire, BCD, Hexadécimal, ASCII)

Durée: 3 heures

Objectifs : Les bases et la numération.
Le codage : la base 2, la base 16, changement de base.
Le code DCB, le code ASCII.
L'arithmétique : addition, nombres signés, soustraction.

Prérequis: Aucun.

Bases théoriques :

Outils :

Supports :

Modalités : Les « * » font référence à des définitions en fin de chapitre, Ctrl + click sur le mot renvoi directement à la définition.

Synthèse et validation :

Ressources existantes :

Travail à réaliser : Exercices en fin de chapitre.

Chaîne d'information

Troisième partie : Codage (Binaire, BCD, Hexadécimal, ASCII)

3.1 – Codage des nombres

3.1.1 - Généralités

L'humanité a mis des millénaires pour passer de la quantité aux nombres. L'idée de nombre est l'aboutissement d'un long travail d'abstraction de la pensée.

Historiquement l'homme compte depuis la préhistoire avec des cailloux et ses doigts. Le mot calcul vient du latin ***calculi*** signifiant caillou. Les civilisations (Grecs, Romains, Chinois, Mayas…) avait développé des systèmes et bases de numération.

Les bases utilisées :

- **Base 5** : quinaire (les doigts de la main), et la **base 20** : visésimal (mains et pieds) utilisées par la civilisation Mayas, elle a été utilisée en France … ? ne dit-on pas « quatre-vingt ».
- **Base 12** : duodécimal, il existe deux organismes la Dozenal Society of America et la Dozenal Society of Great Britain qui font la promotion du système duodécimal en affirmant qu'un système en base 12 est meilleur que le système décimal tant d'un point de vue mathématique que pour d'autres côtés pratiques. En effet 2, 3, 4, 6 sont des diviseurs de 12, ce qui facilite la mise en fraction. Comparé aux diviseurs 2 et 5 du système décimal, le système duodécimal offre plus de possibilités. Il est reconnu que l'observation de 12 lunaisons complètes dans une année explique l'universalité de ce nombre dans toutes les cultures. Des exemples de cet usage sont les 12 mois de l'année, les 12 heures d'une montre, les 12 divisions traditionnelles du temps dans une journée en Chine, les 12 signes du zodiaque de l'astrologie, les 12 signes du zodiaque de l'astrologie chinoise, etc. Il s'utilise encore dans le commerce (douzaine, grosse, etc.). Certaines populations (Moyen-Orient, Roumanie, etc.) connaissent ce système de longue date en comptant les phalanges de la main en omettant celles du pouce (qui est utilisé pour compter les phalanges des autres doigts),ce qui donne bien le chiffre douze, base de cette numération. L'avantage de divisions qui tombent juste explique que les systèmes de mesure aient longtemps comporté des sous-multiples en douzièmes (12 pouces dans un pied, 12 pence dans un shilling, 12 deniers dans un sou, 12 pièces dans une douzaine, 12 douzaines dans une grosse, 12 grosses dans une grande grosse, etc. etc.). À quelques rares exceptions près, dont celle, notable, des États-Unis d'Amérique, ces systèmes ont été abandonnés partout. Le Royaume-Uni a, par exemple, adopté la décimalisation de sa monnaie, la livre sterling depuis 1971.
- **Base 60** Le **système sexagésimal** est un système de numération utilisant la base 60. Au contraire de certaines bases utilisées dans d'autres systèmes (binaire, octal, décimal, hexadécimal), la base 60 n'est actuellement utilisée nulle part de manière systématique. Il subsiste cependant certains vestiges de la numération sexagésimale dans la mesure du temps, des angles et des arcs (y compris les coordonnées géographiques).

Si l'homme est habitué à utiliser le système décimal pour compter(les 10 doigts de la main), les ordinateurs qui fonctionnent en utilisant les propriétés de l'électricité, ne connaissent que les 1 ou les 0. C'est la raison pour laquelle, dès que l'on aborde le monde des systèmes numériques, il est nécessaire pour leur bonne compréhension de connaître certaines bases de numération (binaire, décimale et hexadécimale) et d'être capable d'effectuer des conversions entre-elles. De plus, ces systèmes pour pouvoir communiquer entre-autre (vers un autre système ou vers l'homme) utilisent certains codes (ASCII, UNICODE,…), eux aussi à connaître.

Chaîne d'information

3.1.2 – Numération

Définition : la numération

La **numération** décrit la façon dont les nombres sont représentés. Un système de numération est composé :

- D'un **alphabet** : les signes ou symboles disponibles pour la représentation des nombres
- Des **règles d'écriture** : elles définissent comment un nombre est construit à partir des signes ou symboles de l'alphabet.

Exemple : Le système de numération romaine

L'alphabet : il est composé uniquement de 7 symboles (le 0 n'existe pas dans ce système)

Symbole romain	I	V	X	L	C	D	M
Valeur décimale	1	5	10	50	100	500	1000

Les règles d'écriture :

- Un symbole placé à la droite d'une autre figurant une valeur supérieure ou égale à la sienne s'ajoute à celui-ci.

Ex : VI → donne 6

- Un symbole placé immédiatement à la gauche d'un symbole plus fort que lui, indique que le nombre qui lui correspond doit être retranché au nombre qui suit.

Ex : IV → donne 4

- Les valeurs sont groupées en ordre décroissant, sauf pour les valeurs à retrancher selon la règle précédente.

Ex : CCXLIII → donne 243

- Le même symbole ne peut pas être employé quatre fois consécutivement sauf M.

Ex : le nombre 9 ne s'écrit pas VIIII mais IX.

Remarque : Ce système est très mal adapté pour le calcul.

Exemple 2 : Le système décimal (base 10)

L'alphabet : il est composé de **10** symboles (chiffres) : 0, 1, 2, … 9

Les règles d'écriture :

C'est un système « ***positionnel*** » et ***pondéré***. Chaque position possède un poids : une puissance de 10.

Exemple: le nombre 2895 s'écrit comme 2895=2×1000 + 8×100 + 9×10 + 5×1

Soit sous forme de puissance de 10 : 2895 = $2\times10^3 + 8\times10^2 + 9\times10^1+5\times10^0$ (forme polynomiale)

Littéralement nous l'écrivons : 2895= $a_3xb^3+a_2xb^2+a_1xb^1+a_0xb^0$

Avec: a_3=2, a_2=8, a_1=9, a_0=5, b=10, I = {0, 1, 2, 3}, ***i c'est l'index.***

Le poids correspond à la base à la puissance du rang.

Définition : système décimal

De manière plus formelle dans le système décimal, on pourra écrire tout nombre entier naturel décimal *N* sur *n* chiffres de la manière suivante :

Alphabet : $A_{10} = \{0,1,2,3,4,5,6,7,8,9\}$, ce sont les chiffres utilisés par la base.

$$N_{10} = a_{n-1}a_{n-2}\dots a_2a_1a_0 = \sum_{i=0}^{n-1} a_i \times 10^i = forme\ concaténée\ du\ polynome\ carctéristique$$

(**Σ*** symbole mathématique de la somme)

Avec $a_i \in A_{10}$

- a_i : chiffre de rang *i* correspondant à la i[ème] puissance de 10
- a_0 : chiffre de rang 0 appelé chiffre de poids faible, ***c'est les unités en décimal***.
- a_{n-1} : chiffre de rang *n-1* appelé chiffre de poids fort

Chaîne d'information

Définition : système à base b

On peut étendre la définition précédente à n'importe quel système positionnel pondéré à base *b*. Dans ce système on dispose d'un alphabet A ayant *b* symboles (d_j) : avec j = {0, 1, 2, 3,…, b-1}

Alphabet : $A_b = \{d_0, d_1, d_2, \dots d_{b-2}, d_{b-1}\}$

Le nombre *N* en base *b* s'écrit alors :

$$N_b = a_{n-1}a_{n-2} \dots a_2 a_1 a_0$$

Avec les $a_i \in A_b$

La valeur de N en base 10 est donnée par la relation :

$$N_{10} = \sum_{i=0}^{n-1} a_i \times b^i$$

Avec les a_i en représentation en base 10, i correspond à la valeur du rang.

3.1.3 – Le codage Binaire, Hexadécimal

3.1.31 – Le codage Binaire:

Dans le système binaire, seuls deux symboles permettent d'écrire un nombre : le '0' et le '1'. C'est dans cette base que fonctionnent tous les systèmes électroniques et informatiques qui sont constitués de composants électroniques interprétant des signaux électriques binaires.

Alphabet : $A_2 = \{0,1\}$

Le nombre *N* en base *2* s'écrit alors :

$$N_2 = a_{n-1}a_{n-2} \dots a_2 a_1 a_0$$

Avec les $a_i \in A_2$

Exemple de nombre binaire : $N_2=10010_2$

Convention d'écriture des nombres binaires :

Pour éviter toute confusion lorsque l'on travaille sur des nombres avec des bases différentes, on précise la base dans lequel le nombre est écrit selon les façons suivantes (il n'y a pas de règle précise) :

- Ecriture en binaire : $N_2=(1000110)_2= (1000110)_B$ =%1000110 (en informatique)
- Ecriture en décimal : $N_{10}=(70)_{10}=(70)_D$=70 souvent si c'est du décimal on ne précise pas la base.

Quelques définitions :

Dans la base **binaire**, plutôt que de parler de chiffre, on parle de « **bit** » pour la contraction anglaise de « **bi**narydigi**t** ». **Le poids** (c'est la base à la puissance du rang) du bit est défini par son **rang** (en commençant par 0 pour un entier naturel).

$$N_2 = a_{n-1}a_{n-2} \dots a_2 a_1 a_0$$

Exemple : le bit a_3 d'un nombre binaire N_2 a pour rang 3 et son poids = $(\text{base})^{\text{rang}} = 2^3=8$.

Un **octet** (ou *byte* en anglais) est un nombre codé sur 8 bits. Sur un octet, on va pouvoir coder un entier naturel de 0 à 255. En effet un bit ne pouvant prendre que 2 valeurs (0 ou 1), un octet comporte 8 bits, il y a donc 2^8 combinaisons possibles, soit 256 combinaisons.

D'une manière générale, un nombre binaire sur *n* bits pourra coder un nombre entier naturel $N=2^n-1$. En effet en comptant le 0, il y a 2^n combinaisons.

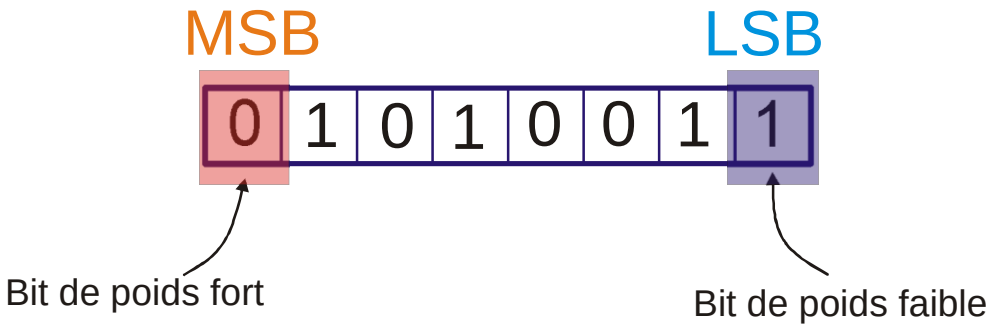
Chaîne d'information

Réciproquement, pour connaître le nombre de bits *n* nécessaire pour coder un nombre entier naturel N, il suffit de calculer :

$$n = \frac{\ln(N+1)}{\ln(2)}$$

Si le résultat n'est pas entier, il faut prendre l'entier juste au-dessus.

LSB : Least Significant Bit = bit de poids faible
MSB : Most Significant Bit = bit de poids fort



Quelques unités :

Pour éviter la confusion, on distingue maintenant les unités ayant un préfixe binaire (CEI :http://fr.wikipedia.org/wiki/Commission_%C3%A9lectrotechnique_internationale) qui représente un facteur de puissance de 2 (encore très utilisé par les électroniciens et les informaticiens mais en utilisant les préfixes SI d'où la confusion) des unités ayant un préfixe décimal en concordance avec le système international (SI) des unités.

Voici à gauche le tableau des préfixes binaires et à droite celui des préfixes SI pour comparaison.

Préfixes binaires (préfixes CEI)

Nom	Sb	Facteur
kibi	Ki	2^{10} = 1 024
mébi	Mi	2^{20} = 1 048 576
gibi	Gi	2^{30} = 1 073 741 824
tébi	Ti	2^{40} = 1 099 511 627 776
pébi	Pi	2^{50} = 1 125 899 906 842 624
exbi	Ei	2^{60} = 1 152 921 504 606 846 976
zébi	Zi	2^{70} = 1 180 591 620 717 411 303 424
yobi	Yi	2^{80} = 1 208 925 819 614 629 174 706 176

Préfixes décimaux (préfixes SI)

Nom	Sb	Facteur	Err.
kilo	k	10^3 = 1 000	2 %
méga	M	10^6 = 1 000 000	5 %
giga	G	10^9 = 1 000 000 000	7 %
téra	T	10^{12} = 1 000 000 000 000	9 %
péta	P	10^{15} = 1 000 000 000 000 000	11 %
exa	E	10^{18} = 1 000 000 000 000 000 000	13 %
zetta	Z	10^{21} = 1 000 000 000 000 000 000 000	15 %
yotta	Y	10^{24} = 1 000 000 000 000 000 000 000 000	17 %

Comptage en binaire :

On compte en binaire comme en décimal, le nombre de symbole étant limité à 2 (0 et 1) l'écriture d'un nombre en binaire nécessitera plus de rang qu'en base 10 (10 symboles, les dix chiffres). Le codage du chiffre 9 en base 10 nécessitera 4 rangs en binaire. $9 = 1001_2 = 1x2^3+0x2^2+0x2^1+1x2^0 = 8 + 1 = 9$ (CQFD).

Rem : $2^0=1$, plus généralement $n^0=1$, tous les nombres à la puissance 0 donne 0 mathématiquement.

Chaîne d'information

Tableau de correspondance (base 10 : base 2)

Décimal	0	1	2	3	4	5	6	7	8	9	10	99	100	1000
Binaire	0	1	10	11	100	101	110	111	1000	1001	1010	1100011	1100100	1111101000

On remarque en binaire que les nombres impairs on le bit de rang 0 à 1, seul poids impair de la base 2.

3.1.32 – Le codage Hexadécimal:

L'écriture d'un nombre binaire est fastidieuse et prompte à de nombreuses erreurs de retranscription. Pour exemple, le nombre décimal 27986, en binaire conduit à écrire le nombre suivant 110110101010010. C'est la raison pour laquelle les informaticiens utilisent la base hexadécimale afin d'avoir une écriture plus concise d'un nombre.

Par exemple, un octet (8 bits) en hexadécimal est codé par deux symboles (sur 2 rangs), un mot de 16 bits par 4 symboles (4 rangs). On peut alors se demander pourquoi ne pas utiliser tout simplement le décimal, mais comme nous le verrons plus tard, la conversion entre base binaire et base hexadécimale est immédiate. Ce qui est loin d'être le cas entre les bases binaire et décimale.

La base hexadécimale comporte 16 symboles. Les 10 premiers sont tout simplement les 10 chiffres. Pour les six restants, ce sont les 6 premières lettres (majuscules) de l'alphabet.

Alphabet : $A_{16} = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Le nombre *N* en base *16* s'écrit alors :

$$N_b = a_{n-1}a_{n-2}\dots a_2a_1a_0$$

Avec les $a_i \in A_{16}$

Exemples de nombre hexadécimal et diverses écritures utilisées :

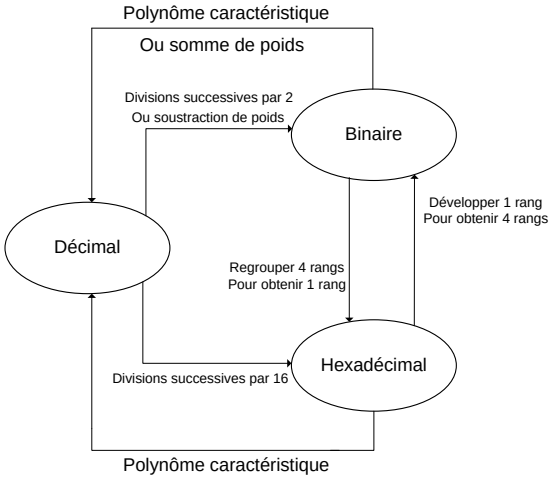
N1= $(1A4)_{16}$; N2= 5023_H ;
N3= 0xA8 (écriture héritée du langage C, langage de programmation, langage évolué)
N4= &H6FE (écriture héritée du basic)
N5= $FFFF (écriture héritée de l'assembleur, langage de programmation, langage machine)

Correspondance Décimal-Binaire-hexadécimal :

Décimal	Binaire	Hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

3.1.4 – Les changements de base

Nous utilisons trois bases (Binaire, Décimal, Hexadécimal), il nous faudra souvent convertir les nombres dans l'une de ces bases, que ce soit du décimal vers le binaire ou l'hexadécimal et inversement du binaire ou de l'hexadécimal vers le décimal. La figure suivante résume les méthodes de conversion entre ces différentes bases.



Conversion d'une Base B (binaire ou hexadécimal ou …) vers la base décimale

Il suffit de reprendre la définition d'un système positionnel pondéré à base b : **le polynôme caractéristique**. *Plus généralement lorsque la base d'arrivée est le décimal on utilisera toujours le polynôme caractéristique et cela quelle que soit la base de départ.*

La valeur de N en base 10 est donnée par la relation :

$$N_{10} = \sum_{i=0}^{n-1} a_i \times b^i$$

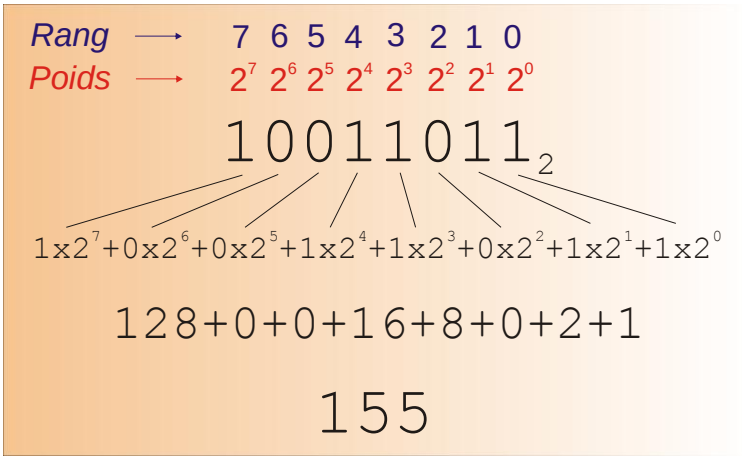
Les a_i étant pris dans leur équivalent décimal.

Application : Binaire → Décimal

$$N_{10} = \sum_{i=0}^{n-1} a_i \times 2^i$$

Par exemple : le nombre binaire $N=10011011_2$ correspond au nombre décimal :

$N = 1.2^7 + 0.2^6+0.2^5+1.2^4+1.2^3+0.2^2+1.2^1+1.2^0$



Chaîne d'information

Application : Décimal → Hexadécimal

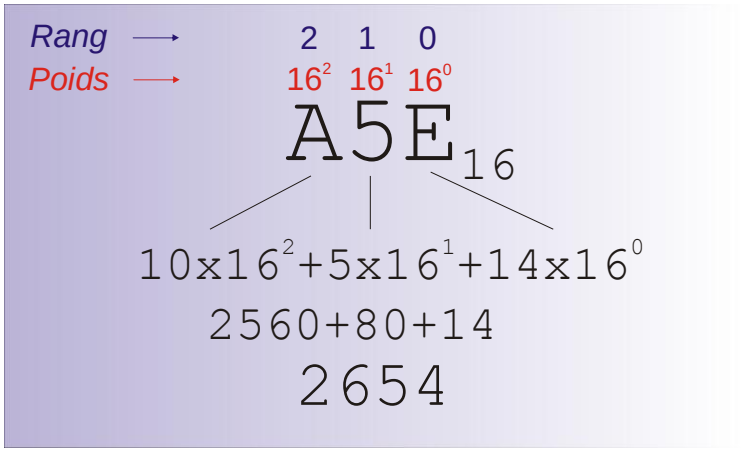
$$N_{10} = \sum_{i=0}^{n-1} a_i \times 16^i$$

Avec les a_i pris dans leur équivalent décimal

Par exemple : le nombre hexadécimal $N=A5E_{16}$ correspond au nombre décimal :

$N = 10\times16^2 + 5\times16^1 + 14\times16^0$

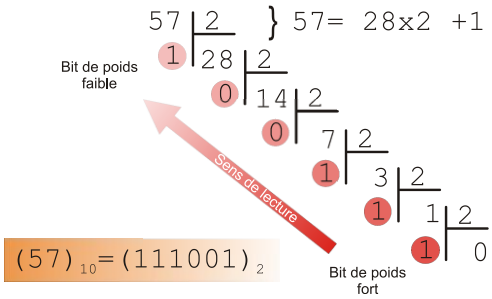
Soit $N= 10\times256 + 5\times16 + 14 = 2560 + 80 + 14 = 2654$



Conversion décimale vers le binaire ou l'hexadécimal

Pour convertir un nombre décimal dans une autre base (binaire, hexadécimal, ou autre) il faut le diviser par la base d'arrivée. On effectue des divisions entières successives jusqu'à ce que le quotient soit nul.

Exemple : décimal en binaire



Cas particulier du binaire : les symboles de la base 2 étant « **0** » et « **1** », le polynôme caractéristique est équivalent à une somme de poids. Si le symbole vaut « **1** » on prendra le poids du rang sinon c'est $0\times2^i=0$ pour le rang i. On peut donc pour convertir un nombre décimal en binaire en effectuant une soustraction de poids.

Méthode : On soustrait au nombre décimal le poids binaire le plus grand possible afin de déterminer le rang à 1 le plus élevé, puis on recommence avec le reste de la soustraction et cela jusqu'à trouver zéro comme reste. Cette méthode permet de convertir les nombres plus rapidement que la division successive qui reste rébarbative en binaire.

Chaîne d'information

Les poids en binaire :

Rang	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Poids	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Décimal	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536

Reprenons l'exemple du nombre 57

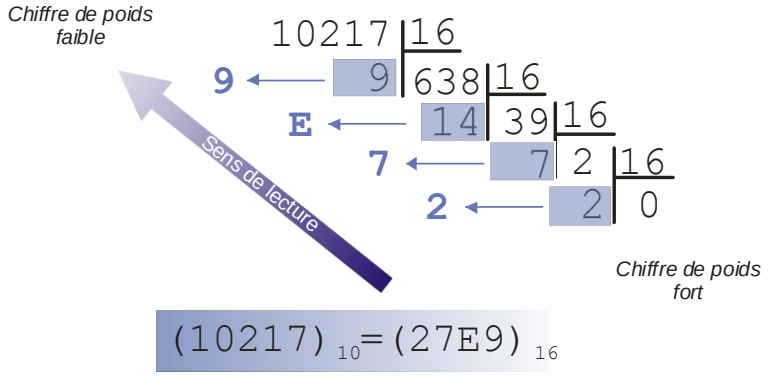
On obtient les opérations suivantes : $57 - 32(2^5) = 25$, rang 5=1 en binaire

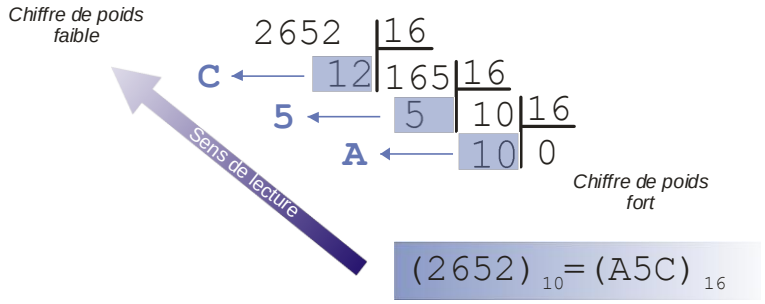
$25 - 16(2^4) = 9$, rang 4=1

$9 - 8(2^3) = 1$, rang 3=1

$1 - 1(2^0) = 0$, rang 0=1 donc $57 = 111001_2$

Exemple : décimal en hexadécimal





Conversion binaire → hexadécimal et inversement

Mathématiquement c'est le passage d'une base B dans une base B^i, s'il existe une relation entre la base de départ et la base d'arrivée (ici B) on pourra regrouper i rangs de la base de départ pour obtenir un rang dans la base d'arrivée. Quand on passe de la base 2(binaire) à la base 16(hexadécimal) on est bien dans ce cas là puisque si B=2 alors $16=2^4=B^4$. Dans cette conversion i= 4, cela nous permet de regrouper 4 rangs en binaire pour obtenir un rang en hexadécimal. On effectue les regroupements pour un nombre entier en partant du rang « 0 » vers les rangs les plus élevés. Lorsque que l'on fera l'opération inverse, passer de l'hexadécimal vers le binaire (base B^i(16)vers B(2)), on pourra développer chaque rang de la base hexadécimal en 4 rangs en binaire.

Exemple : Binaire en Hexadécimal

Rang	15……..……....0
Binaire	1001110010100111
Hexa	9 C A 7

Exemple : Hexadécimal en Binaire

Hexa	8 6 5 B
Binaire	1000011001011011
Rang	15……..……....0

3.1.5 – Les codes DCB(BCD) et ASCII

3.1.51 – Les codes DCB(BCD)

L'abréviation **DCB** signifie ***D**écimal **C**odé en **B**inaire* (ou code **BCD** en anglais pour ***B**inary **C**oded **D**ecimal)*. C'est un codage utilisé en électronique numérique pour la commande de circuits d'IHM (Interfaçage Homme-Machine). Historiquement les IHM étaient relativement rudimentaires, pour traduire les différentes informations (numérique) d'un système on utilisait (ça existe encore …) des afficheurs 7 segments (1 afficheur par chiffre) dont les circuits de commandes nécessités une valeur DCB en entrée. L'évolution technologique nous fournit aujourd'hui des afficheurs intelligents (graphiques) dont la commande est gérée par un bus de communication (série ou parallèle). Les calculateurs possèdent toujours un jeu d'instruction capable d'effectuer des opérations en binaire naturel ou en DCB. Le codage DCB traduit chaque chiffre décimal en un mot binaire codé sur 4 rangs, en binaire naturel sur 4 rangs on a 16 possibilités (16 valeurs différentes de 0.. à ..15), on en utilisera 10 de ces possibilités pour coder les 10 chiffres du décimal. On pourra remarquer que de coder les nombres en BCD augmentera l'occupation mémoire par rapport au binaire naturel.

Correspondance Décimal-Binaire-DCB :

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

Chaîne d'information

Exemples :

Le nombre décimal (5836) est codé en BCD par : $(0101\ 1000\ 0011\ 0110)_{BCD}$
Le code $(0111\ 0100\ 0000\ 0010)_{BCD}$ correspond au nombre décimal : $(7402)_{10}$
Ce code (0101 1000 1011 0010) n'est pas un code BCD car la combinaison 1011 n'existe pas.

Rem : Les données sont manipulées par les calculateurs en mot de 8bits (1 byte, 1octet), on pourra donc stocker sur un octet l'équivalent de deux chiffres codés en BCD, chaque chiffre occupant 4 bits (un quartet).

3.1.52 – Les codes ASCII*

Le code **ASCII** a été créé en 1961 par Bob Berner et permettait au départ d'afficher 128 caractères. ASCII (7 bits en binaire), voulant dire "**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange", est donc un standard de codage des caractères.

Pourquoi un codage sur 7/8 bits ?

Pour l'édition de documents alphanumériques simples dans la civilisation occidentale. Nous avons besoin de :

- Un alphabet de lettres minuscules = {a, b, c,..., z} soient 26 caractères
- Un alphabet de lettres majuscules = {A, B, C,..., Z} soient 26 caractères
- Des chiffres {0, 1, ..., 9} soient 10 caractères
- Des symboles syntaxiques {?, ; (" …} au minimum 10 caractères

Soit un total minimal de 72 caractères.

Si on choisit un code à 6 bits le nombre de caractères que l'on peut coder est de $2^6 = 64$ (de 000000_2 à 111111_2), insuffisant donc pour nos besoins. Il faut au minimum 1 bit de plus, ce qui permet de définir ainsi $2^7 = 128$ nombres binaires différents, autorisant alors le codage de 128 caractères. Initialement le code ASCII est un code à 7 bits ($2^7 = 128$ caractères). Il a été étendu à un code sur 8 bits ($2^8 = 256$ caractères) permettant le codage des caractères nationaux (en France les caractères accentués comme : ù, à, è, é, â, etc.) et les caractères semi-graphiques.

Le code **ASCII** a ensuite été étendu par l'
Unicode* pour pouvoir représenter l'ensemble des caractères utilisés dans les différents pays. La première version d'Unicode date de 1991. Le codage Unicode se faisait sur 16 bits et est passé à 32 bits. Il inclut tous les alphabets connus, la dernière version Unicode 6.1.0 a été publiée le 31 janvier 2012.

3.2 – Arithmétique Binaire

3.2.1 –Addition

Quand vous faites une addition en décimal, vous faites la somme des chiffres se trouvant dans une même colonne. Si la somme est inférieure à 10, alors vous posez le résultat obtenu et passez à la colonne suivante. Si la somme est supérieure à 10, alors vous posez le chiffre des unités et gardez en retenue le chiffre des dizaines. Si vous faites la somme de 2 nombres, alors la retenue ne pourra être supérieure à 1 (9+9=18).

Le principe est exactement le même en binaire.

Vous faites la somme, posez le chiffre des unités, et retenez le chiffre de la seconde colonne en retenue (qu'il vaut mieux, évidemment, éviter d'appeler les« dizaines »).

Si vous faites la somme de **2 nombres**, alors il n'y a que **4 cas possibles** :

- 0 + 0 = 0, on pose 0
- 0 + 1 = 1, on pose 1
- 1 + 1 = 10_2, on pose 0 et on retient 1
- 1 + 1 + une retenue de 1 = 11_2, on pose 1 et on retient 1

Exemple : ***19 + 71***

19 est codé en binaire 10011_2 = 16+2+1=19, codé sur 5 rangs (du rang 0 au rang 4).

71 est codé en binaire 1000111_2=64+4+2+1=71, codé sur 7 rangs (rang 0 au rang6).

Pour effectuer cette addition il faut utiliser un format identique pour coder ces deux nombres. Les données étant manipulées par octet, on codera sur 8 rangs les deux nombres, il suffit de rajouter des 0 sur les rangs supérieurs, on obtient :

19 est codé en binaire sur 1 octet 00010011_2 = 13_H

71 est codé en binaire sur 1 octet 01000111_2 = 47_H

On obtient :

Retenue →								
19	0	0	0	1	0	0	1	1
+71	0	1	0	0	0	1	1	1
=								

Rem : Si les nombres sont supérieurs à la valeur maximale sur un octet (soit supérieur à 255), on les codera sur deux, trois,… octets, autant que nécessaire. Ex: 65535 ($FFFF_H$) sera codé sur 2 octets mais 65536 sur trois.

3.2.2– Nombre signé

En arithmétique signé un calculateur doit pouvoir stocker en mémoire des nombres négatifs. Pour représenter un nombre négatif dans sa mémoire le calculateur a besoin d'un bit de signe. Le bit de signe d'un nombre signé sera toujours représenté par le MSB, bit de rang le plus élevé. Il faut donc ajouter un bit supplémentaire au codage du nombre au rang immédiatement supérieur. Pour effectuer les calculs sur des nombres signés le calculateur les manipulant par octet, il faudra souvent coder les nombres signés par multiple de 8bits. Ex : 255 = FF_H = $1111\ 1111_2$, si on veut coder -255 il faudra travailler sur un format 16bits.

Si le bit de signe = 0, le nombre est positif.

Si le bit de signe = 1, le nombre est négatif.

Pour transformer un nombre binaire positif en nombre négatif, il faut effectuer **le complément à deux** de ce nombre, cette opération consiste à complémenter (transformer les 0 en 1 et les 1 en 0) tous les bits du nombre (c'est le complément à 1, ou complément restreint) et à lui ajouter 1 (on obtient le complément à 2 ou complément arithmétique ou encore complément vrai).

N1 complémenté (complément à 1) s'écrit $\overline{N1}$

Le complément à 2 de N1 s'écrit $\overline{N1} + 1$

Donc $\overline{N1} + 1 = -N1$

Chaîne d'information

Exemple :

Bit de signe = 0, positif

N1 = 71 = % 0100 0111

$\overline{N1}$= % 1011 1000 ← Complément à 1

\+ 1

$\overline{N1}$ + 1 = % 1011 1001 ← Complément à 2

Bit de signe = 1, négatif

Rem : il existe une méthode pour transformer un nombre positif en négatif directement. Pour obtenir le complément à 2 d'un nombre binaire, on parcourt le nombre en partant du rang 0 vers les rangs plus élevés, dès que l'on rencontre un 1, on complémente les bits de rang supérieur.

Bit de signe = 0, positif

N1 = 71 = % 0100 0111 ← J'ai trouvé Le premier 1

-N1 = % 1011 1001

Bit de signe = 1, négatif

La représentation binaire d'un nombre négatif en mémoire n'est pas directement représentative de la valeur de ce nombre, pour obtenir la valeur d'un nombre négatif il faut effectuer à nouveau le complément à 2 du nombre négatif (cette opération inverse le signe du nombre, soit – (-N1)=N1) pour obtenir son équivalent positive, il suffit de transformer ce nombre binaire en décimal pour obtenir sa valeur.

Il existe une autre méthode qui permet de traduire directement un nombre binaire négatif en son équivalent décimal positif, il suffit de considérer que le poids du bit de signe est un poids négatif, les autres poids du nombre restants positifs.

Dans notre exemple cela donne : -128 + 32 + 16 +8 + 1 = -71 (CQFD)

3.2.3–Soustraction

Un calculateur ne sait pas soustraire au sens arithmétique, par contre il sait inverser le signe d'un nombre en arithmétique signé nous venons de le voir. Pour réaliser une soustraction N1 – N2, le calculateur vas transformer le signe du nombre N2 pour obtenir –N2, il effectuera la soustraction en additionnant N1 + (-N2).

Lorsque que l'on réalise une soustraction, on travaille sur un nombre de bit constant, lorsque le résultat est positif, il provoque l'éjection d'un bit à 1 au rang supérieur au bit de signe.

Exemple : 19 – 71

11 11

N1 = 19 = % 0001 0011

N2 = 71 = % 0100 0111 +

-N2 = % 1011 1001

R = % 1100 1100

Bit de signe = 1, négatif

R = -128 + 64 + 8 + 4 = -52

Chaîne d'information

3.3 – Exercices

3.3.1 – Codage

Compléter le tableau suivant en effectuant les conversions :

Décimal	Binaire (8 ou 16bits)	Hexadécimal (8 ou 16bits)
108		
	0000000101100001	
		C2
	0000001000011111	
		04BD
2058		

3.3.2 – Arithmétique

1 Addition non signé:

- Réalisez les additions suivantes dans la base correspondante.

	1001 1011
+	0001 1111
=	

	0101 0001
+	0101 0001
=	

2 Représentation des nombres signés:

- Traduisez les nombres suivants en binaire sur 8 bits.

-15=

-23 =

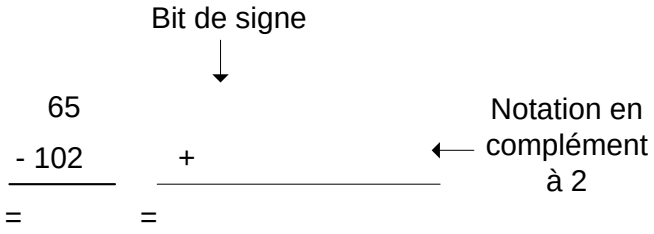
-125 =

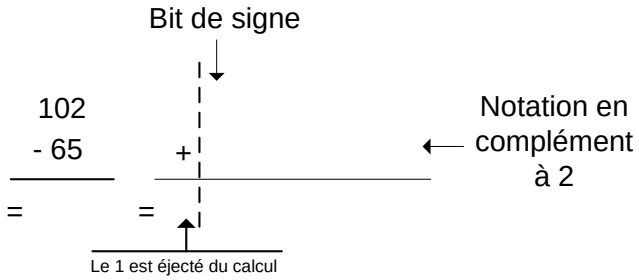
-10 =

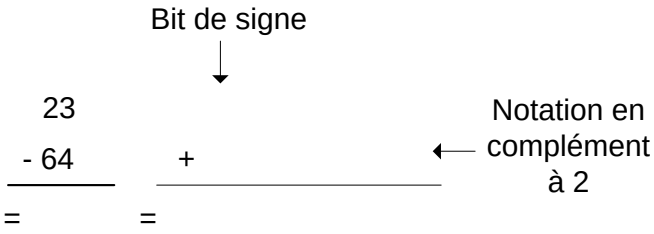
Chaîne d'information

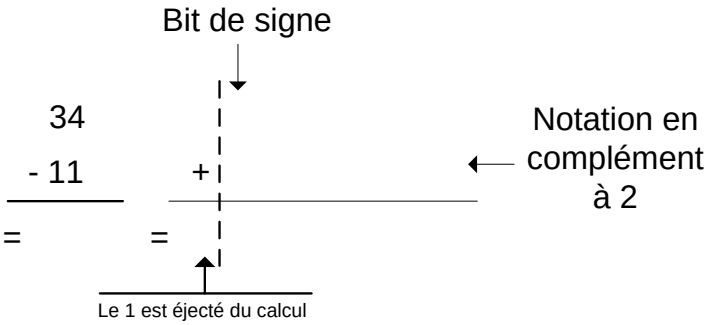
3 Soustraction:

- Réalisez les soustractions suivantes en BINAIRE.









Chaîne d'information

Lexique :

Σ : (sigma) symbole mathématique de la somme, pour exprimer une longue série d'additions de terme récurent. On le rencontre souvent dans l'expression d'un polynôme.

Exemples :http://fr.wikipedia.org/wiki/Notation_(math%C3%A9matiques)

- Si n est un entier strictement positif :

$$\sum_{k=1}^{n} k^2 = 1^2 + 2^2 + 3^2 + 4^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Ici k est la variable muette, elle prend ses valeurs dans l'ensemble [1,n] (ensemble d'entiers). Le terme général de cette somme est k^2.

ASCII : table des caractères ASCII standard.

Dec	Hex	Character	Dec	Hex	HTML	Char	Dec	Hex	HTML	Char	Dec	Hex	HTML	Char
000	00	NUL (null)	032	20	&#32;	Space	064	40	&#64;	@	096	60	&#96;	`
001	01	SOH (start of header)	033	21	&#33;	!	065	41	&#65;	A	097	61	&#97;	a
002	02	STX (start of text)	034	22	&#34;	"	066	42	&#66;	B	098	62	&#98;	b
003	03	ETX (end of text)	035	23	&#53;	#	067	43	&#67;	C	099	63	&#99;	c
004	04	EOT (end of transmission)	036	24	&#36;	$	068	44	&#68;	D	100	64	&#100;	d
005	05	ENQ (enquiry)	037	25	&#37;	%	069	45	&#69;	E	101	65	&#101;	e
006	06	ACK (acknowledgement)	038	26	&#38;	&	070	46	&#70;	F	102	66	&#102;	f
007	07	BEL (bell)	039	27	&#39;	'	071	47	&#71;	G	103	67	&#103;	g
008	08	BS (backspace)	040	28	&#40;	(	072	48	&#72;	H	104	68	&#104;	h
009	09	TAB (horizontal tab)	041	29	&#41;	)	073	49	&#73;	I	105	69	&#105;	i
010	0A	LF (line feed)	042	2A	&#42;	*	074	4A	&#74;	J	106	6A	&#106;	j
011	0B	VT (vertical tab)	043	2B	&#43;	+	075	4B	&#75;	K	107	6B	&#107;	k
012	0C	FF (form feed)	044	2C	&#44;	,	076	4C	&#76;	L	108	6C	&#108;	l
013	0D	CR (carriage return)	045	2D	&#45;	-	077	4D	&#77;	M	109	6D	&#109;	m
014	0E	SO (shift out)	046	2E	&#46;	.	078	4E	&#78;	N	110	6E	&#110;	n
015	0F	SI (shift in)	047	2F	&#47;	/	079	4F	&#79;	O	111	6F	&#111;	o
016	10	DLE (data link escape)	048	30	&#48;	0	080	50	&#80;	P	112	70	&#112;	p
017	11	DC1 (device control 1)	049	31	&#49;	1	081	51	&#81;	Q	113	71	&#113;	q
018	12	DC2 (device control 2)	050	32	&#50;	2	082	52	&#82;	R	114	72	&#114;	r
019	13	DC3 (device control 3)	051	33	&#51;	3	083	53	&#83;	S	115	73	&#115;	s
020	14	DC4 (device control 4)	052	34	&#52;	4	084	54	&#84;	T	116	74	&#116;	t
021	15	NAK (negative acknowledge)	053	35	&#53;	5	085	55	&#85;	U	117	75	&#117;	u
022	16	SYN (synchronous idle)	054	36	&#54;	6	086	56	&#86;	V	118	76	&#118;	v
023	17	ETB (end of trans. block)	055	37	&#55;	7	087	57	&#87;	W	119	77	&#119;	w
024	18	CAN (cancel)	056	38	&#56;	8	088	58	&#88;	X	120	78	&#120;	x
025	19	EM (end of medium)	057	39	&#57;	9	089	59	&#89;	Y	121	79	&#121;	y
026	1A	SUB (substitute)	058	3A	&#58;	:	090	5A	&#90;	Z	122	7A	&#122;	z
027	1B	ESC (escape)	059	3B	&#59;	;	091	5B	&#91;	[	123	7B	&#123;	{
028	1C	FS (file separator)	060	3C	&#60;	<	092	5C	&#92;	\	124	7C	&#124;	|
029	1D	GS (group separator)	061	3D	&#61;	=	093	5D	&#93;	]	125	7D	&#125;	}
030	1E	RS (record separator)	062	3E	&#62;	>	094	5E	&#94;	^	126	7E	&#126;	~
031	1F	US (unit separator)	063	3F	&#63;	?	095	5F	&#95;	_	127	7F	&#127;	DEL

Unicode

Unicode est une norme informatique, développée par le *Consortium Unicode*, qui vise à permettre le codage de texte écrit en donnant à tout caractère de n'importe quel système d'écriture un nom et un identifiant numérique, et ce de manière unifiée, quelle que soit la plate-forme informatique ou le logiciel. La dernière version, **Unicode 6.1.0**, est publiée depuis le 31 janvier 2012.

Totalement compatible avec le jeu universel de caractères (JUC) de l'ISO/CEI 10646, la norme Unicode l'étend en lui ajoutant un modèle de représentation et de traitement de textes complets, en conférant à chaque caractère un jeu de propriétés normalisées ou informatives, en décrivant avec précision les relations sémantiques qui peuvent exister entre plusieurs caractères successifs d'un texte, et en normalisant des algorithmes de traitement qui préservent au maximum la sémantique des textes transformés, tout en étendant l'interopérabilité de la représentation de ces textes sur des systèmes hétérogènes.

Le standard Unicode est constitué d'un répertoire de plus de 109 000 caractères couvrant 93 écritures, d'un ensemble de tableaux de codes pour référence visuelle, d'une méthode de codage et de plusieurs codages de caractères standard, d'une énumération des propriétés de caractère (lettres majuscules, minuscules, symboles, ponctuation, etc.) d'un ensemble de fichiers de référence des données informatiques, et d'un certain nombre d'éléments liés, tels que des règles de normalisation, de décomposition, de tri, de rendu et d'ordre d'affichage bidirectionnel (pour l'affichage correct de texte contenant à la fois des caractères d'écritures de droite à gauche, comme l'arabe et l'hébreu, et de gauche à droite).

En pratique, Unicode reprend intégralement la norme ISO/CEI 10646, puisque cette dernière ne normalise que les caractères individuels en leur assignant un nom et un numéro normatif (appelé *point de code*) et une description informative très limitée, mais aucun traitement ni aucune spécification ou recommandation pour leur emploi dans l'écriture de langues réelles, ce que seule la norme Unicode définit précisément. L'ISO/CEI 10646 fait normativement référence à certaines partie du standard Unicode (notamment l'algorithme bidirectionnel et les propriétés des caractères) ; Unicode est également une norme *de facto* (de fait) pour le traitement du texte et sert de base à de nombreuses autres normes.