Numération et représentation des nombres

Définition :

3 - Représentation de l'information
3.3 - Numération et représentation des nombres
- Système de numération
  * base 10; base 2; base 16.
  * représentation hexadécimale des nombres binaires.
- Représentations codées binaires : binaire pur, BCD, GRAY, ASCII.

Objectifs :

Vous devez être capable à la fin de ce T.P à partir d'un nombre donné codé, de transcrire ce nombre codé dans n'importe quelle base, base 2, base 10, base 16 Puis d'effectuer des opérations en base 2. Eventuellement effectuer des opérations logiques en base 2.

Ce TP est autocorectif.
Vous me demanderez la correction une demie heure avant la fin du TP.

NUMERATION

Bases de numération :

De tout temps, l'homme a cherché à compter avec plus ou moins de réussite. Dès la préhistoire, on retrouve sur des os des entailles pouvant avoir servi à compter des animaux ou des objets. S'il est relativement simple de compter avec des traits jusqu'à 5, il devient plus difficile de compter au-delà. Imaginez "100" représenté avec des entailles, l'erreur est assurée... Les romains ont mis en place un système de numération basé sur des symboles littéraux. Cette numération très utile à cette époque présente de nos jours l'inconvénient majeur de ne pas pouvoir autoriser les opérations de base (addition, soustraction). Essayez d'ajouter de tête MCDIV + CCLXXI. Problème n'est-ce pas ? La réponse est : MDCLXXV (1675).

C'est vers l'an 750 en Inde, que les indiens ont mis en place la numération que nous utilisons aujourd'hui avec tous ses chiffres. Les Indiens les ont ensuite transmis aux Arabes qui les ont transmis aux Européens vers l'an 1200. Et ce n'est qu'environ trois siècles plus tard que les chiffres se généraliserons, un peu. C'est pour cette raison que l'on parle de chiffres arabes. Si les Arabes ne sont pas à l'origine des chiffres, ils sont par contre à l'origine des mathématiques. C'est l'astronome et mathématicien musulman Al Khuwârizmî qui établira les fondements des règles du calcul algébrique.

La numération fait appel à deux principes fondamentaux que l'on retrouvera dans toutes les bases de calcul.

Le premier principe fondamental est le principe de position, on associe à un chiffre qui a une position dans un nombre une valeur parfaitement définie. Par exemple 1 signifie que l'on a **une fois l'unité**. Le même chiffre placé dans le nombre 1254 signifie qu'il y a **une fois mille unités**.

Le deuxième principe fondamental est le principe du zéro. Le zéro matérialise une position où il y absence d'éléments. Pour que le nombre 10 signifie dix, il faut placer le un sur la colonne des dizaines et matérialiser l'absence d'unité par un zéro. Le zéro donne son sens au nombre en positionnant le 1 dans la colonne des dizaines.

Les bases de calcul sont nombreuses même si peu sont réellement utilisés. On peut citer un certain nombre de bases qui ont eu ou qui ont toujours une très grande importance :

- La base 2 (0, 1), ou base binaire, est très utilisée en automatique et informatique industrielle. Elle est la base de la logique booléenne ou algèbre de Boole.
- la base 4 (0, 1, 2, 3) a été l'objet de nombreuses polémiques et aurait pu devenir la base universelle.
- la base 5 (0, 1, 2, 3, 4) est une base très intéressante qui permet facilement de compter jusqu'à 30 avec ses dix doigts. La première main comptant les unités et la deuxième comptant les cinquaines.
- la base 8 (0, 1, 2, 3, 4) ou base octale, a été utilisée il y a un certain temps en informatique lorsque les machines utilisées étaient peu gourmande en puissance de bus.
- la base 10 (base décimale). Elle est considérée comme la base universelle. de ce fait nous l'utilisons presque tout le temps, mais sans vraiment savoir comment elle fonctionne...
- la base 12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, α, β) est une base religieuse établie sur les douze signes du zodiaque. Cette base a été utilisée principalement dans le commerce (c'est à cause de l'utilisation de cette base que l'on parle encore d'une douzaine d'œufs ou d'une douzaine d'huîtres, dans une journée il y a le jour et la nuit chacun divisé en douze heures, etc...)
- la base 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) ou base hexadécimale est très utilisée dans le monde de la micro-informatique et des automates. Elle permet de coder un mot (16 bits) sur 4 variables hexadécimales. Cette base fait intervenir tous les chiffres puis les 6 premières lettres de l'alphabet.
- la base 20. Cette base a été construite à partir des dix doigts des mains et des pieds de l'homme. Cette base a été très utilisée et il en subsiste quelques traces dans notre numération actuelle (quatre-vingts, quatre-vingt-dix).

- la base 60. Cette base a été construite sur des concepts religieux, d'un maniement complexe elle est toujours utilisée de nos jours. (Sur un cercle, il y a 360° qui sont divisés chacun en 60 minutes divisées chacune en 60 secondes. Il en va de même pour chacune des heures de la journée qui sont divisées chacune en 60 minutes, elles-mêmes divisées en 60 secondes. La division des secondes en soixantième de seconde n'ayant pas de sens, la précision se fait en centième de seconde. Il subsiste quelques traces de cette base dans notre numération actuelle (soixante, soixante-dix)).

En informatique industrielle, chaque signal n'ayant que deux états possibles, état 0 et état 1, les bases que nous seront amenés à utiliser pour coder des signaux seront des multiples de la base binaire (Binaire, Octale et Hexadécimale). L'objet de cette étude portera plus particulièrement sur les problèmes de changement de base (10) en base (2) ou (16) ou l'inverse, ainsi que le passage de base (10) en base (16).

Codage en base 2	→	0; 1.
Codage en base 8	→	0;...; 7.
Codage en base 10	→	0;...; 9.
Codage en base 16	→	0; ..., 9, A,...; F.

Définitions :

Bit : Contraction des mots anglais BInary DigiT (traduit par : "chiffre binaire").

C'est l'unité élémentaire d'information qui ne peut prendre qu'une des deux valeurs suivantes : 0 ou 1. Dans les micro-ordinateurs, les bits sont groupés par huit pour former des mots. Un mot de huit bits est un octet ou en Anglais un byte. Quelquefois, on ajoute à un groupe de bits un bit supplémentaire qui sera un bit de contrôle (en anglais : check bit). Si le bit de contrôle forme avec le nombre total de bits un chiffre impair, on parlera de bit d'imparité (en anglais : odd parity), s'il forme un nombre pair, on parlera de bit de parité (en anglais : parity bit). Les microprocesseurs qui utilisent des mots de 8, 16, 32 bits sont appelés microprocesseurs 8 bits, 16 bits, etc.

Octet : C'est un ensemble de huit bits. On exprime généralement la capacité des mémoires en Kilo-octets (Ko), un Ko vaut 1024 octets (2^{10}).

Les micro-ordinateurs ont une capacité de mémoire disponible (RAM : Read Only Memory) égale ou supérieure à 512 Ko (Kilo Octet). Sur les micro-ordinateurs équipés de processeurs rapides, il est fréquent de trouver 128 Mo (Méga Octet) de mémoire vive.

Les disquettes ont une capacité qui varie, selon les modèles, de 180 Ko à 1.44 Mo, voire plus parfois.

Les disques durs peuvent stocker plusieurs centaine, voire plusieurs milliers de Mo. Certains systèmes peuvent même disposer de plusieurs Giga-octets de mémoire de masse. On exprime généralement la capacité des mémoires de masse en Mega-octets (Mo) et giga-octets (Go). Un Mo vaut 1000 Ko. Un Giga-octet vaut 1000 Mo.

Byte : (français : octet).

Attention donc à ne pas confondre un bit et un byte, un byte c'est 8 bits.

Mot : c'est un ensemble de 2 octets, il est donc codé sur 16 bits. (Anglais : Word)

Mot long : c'est deux mots donc 32 bits. (Anglais : Dword pour double word)

Codage parallèle Décimal, Binaire, Octal, Hexadécimal :

Décimal	Binaire	Octal	Héxadécimal
00	0000	00	00
01	0001	01	01
02	0010	02	02
03	0011	03	03
04	0100	04	04
05	0101	05	05
06	0110	06	06
07	0111	07	07
08	1000	10	08
09	1001	11	09
10	1010	12	0A
11	1011	13	0B
12	1100	14	0C
13	1101	15	0D
14	1110	16	0E
15	1111	17	0F
16	10000	20	10

Codage binaire :

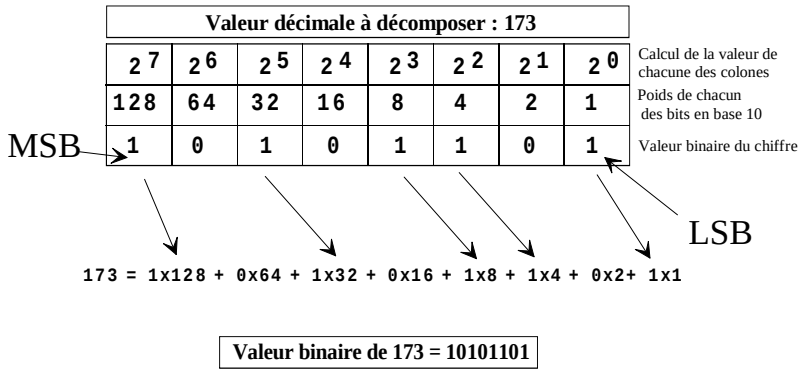
On associe à chaque bit un poids. Ce poids est fonction de la position du bit dans l'octet.

LSB :(Least significant Bit) bit de poids faible, c'est le bit qui a le moins de signification dans un octet. Par convention, c'est le bit le plus à droite dans l'écriture d'un mot. Exactement comme dans le nombre 105, c'est le chiffre le plus à droite qui à le moins de poids, la valeur la plus faible.

MSB :(Most significant bit) bit de poids fort, c'est le bit qui a le plus de signification dans un octet. Par convention, c'est le bit le plus à gauche dans l'écriture d'un mot. Exactement comme dans le nombre 105, c'est le chiffre le plus à gauche qui à le plus de poids, la valeur la plus forte.

Explication par l'exemple : l'octet 1010 1101 représente le nombre 173.

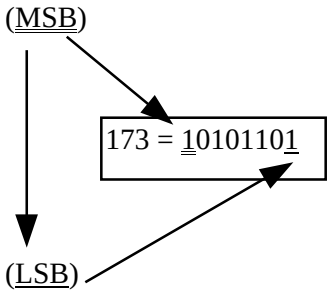
Décomposition :



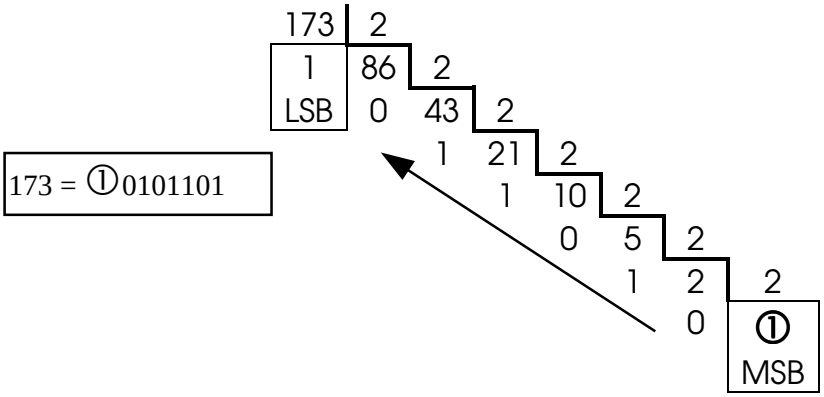
Valeur binaire de 173 = 10101101

Méthode de la soustraction (méthode un peu lourde):

Dans 173, y-a-t-il 128 ? oui 173 -128 = 45 → 1
Dans 45, y-a-t-il 64 ? non, il reste toujours 45 → 0
Dans 45, y-a-t-il 32 ? oui 45 - 32 = 13 → 1
Dans 13, y-a-t-il 16 ? non, il reste toujours 13 → 0
Dans 13, y-a-t-il 8 ? oui 13 - 8 = 5 → 1
Dans 5, y-a-t-il 4 ? oui 5 - 4 = 1 → 1
Dans 1, y-a-t-il 2 ? non, il reste toujours 1 → 0
Dans 1, y-a-t-il 1 ? oui 1 - 1 = 0 → 1



Méthode de la division (méthode élégante) :



Le principe de la méthode par divisions successives consiste à réaliser une suite de divisions par 2. Chaque quotient devient à son tour dividende jusqu'à obtenir un quotient égal à 1. Les restes de ces divisions sont toujours 0 ou 1 puisque les nombres à diviser sont soit pairs soit impairs. Le dernier quotient obtenu (1) est le MSB. Il faut donc remonter les restes successifs du MSB au LSB pour obtenir la valeur binaire d'un nombre entier jusqu'au LSB.

Codage Hexadécimal :

Le principe de base est le même que pour le codage binaire. Mais dans ce cas, la base de comptage est la base 16.

Valeur décimale = 173

16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
•••	•••	•••	65536	4096	256	16	1
0	0	0	0	0	0	A	D

$$173_{(10)} = 10_{(10)} x 16_{(10)} + 13_{(10)} x 1_{(10)}$$
$$AD_{(hex)} = A_{(hex)} x 16_{(10)} + D_{(hex)} x 1_{(10)}$$

Valeur héxadécimale = AD

Ecriture du codage binaire décimal et hexadécimal:

8	4	2	1	
0	0	0	0	Binaire
0				Décimal
0				Héxadécimal

8	4	2	1	
0	1	0	0	Binaire
4				Décimal
4				Héxadécimal

8	4	2	1	
1	0	0	0	Binaire
8				Décimal
8				Héxadécimal

8	4	2	1	
1	1	0	0	Binaire
12				Décimal
C				Héxadécimal

8	4	2	1	
0	0	0	1	Binaire
1				Décimal
1				Héxadécimal

8	4	2	1	
0	1	0	1	Binaire
5				Décimal
5				Héxadécimal

8	4	2	1	
1	0	0	1	Binaire
9				Décimal
9				Héxadécimal

8	4	2	1	
1	1	0	1	Binaire
13				Décimal
D				Héxadécimal

8	4	2	1	
0	0	1	0	Binaire
2				Décimal
2				Héxadécimal

8	4	2	1	
0	1	1	0	Binaire
6				Décimal
6				Héxadécimal

8	4	2	1	
1	0	1	0	Binaire
10				Décimal
A				Héxadécimal

8	4	2	1	
1	1	1	0	Binaire
14				Décimal
E				Héxadécimal

8	4	2	1	
0	0	1	1	Binaire
3				Décimal
3				Héxadécimal

8	4	2	1	
0	1	1	1	Binaire
7				Décimal
7				Héxadécimal

8	4	2	1	
1	0	1	1	Binaire
11				Décimal
B				Héxadécimal

8	4	2	1	
1	1	1	1	Binaire
15				Décimal
F				Héxadécimal

Sur cet ensemble de tableaux on voit qu'il est donc nécessaire pour coder complètement tous les termes de coder chacune des variables hexadécimales sur 4 bits.

La valeur hexadécimale suivant $F_{(16)}$ sera $16_{(10)}$ se codera donc $10_{(16)}$ et $0001\ 0000_{(2)}$.

Représentation des nombres négatifs :

Avec une représentation numérique construite uniquement avec des 1 et des 0 pas question de signes + ou -. Il faut trouver un artifice où les symboles + et - n'interviennent pas. La représentation des valeurs doit donc se faire exclusivement par des valeurs binaires.

* Le complément vrai (CPLV):

Le complément vrai ou complément à b (b signifiant la base dans laquelle on travaille) par rapport à un radical N s'obtient en faisant la soustraction A complémenté = N - A, avec N>A.

Ainsi le complément vrai en base 10 de 529 par rapport à 1000 est 471, par rapport à 10000 est de 9471.

L'opération consiste à complémenter chacun des bits d'une variable binaire.
Exemple : Le complément vrai de l'octet $16_{(16)}$.

$$16_{(16)} = 0001\ 0110_{(2)}$$

Son complément vrai sera :

$$\text{CPLV } 16_{(16)} = 1110\ 1001_{(2)} = E9_{(16)}$$

Remarque : on peut arriver plus rapidement à ce résultat en faisant

$$FF_{(16)} - 16_{(16)} = E9_{(16)}$$

* Complément à 2 (Négation) (CPL2):

L'opération consiste à complémenter chacun des bits d'une variable binaire et à ajouter 1.

exemple : la négation de l'octet $27_{(16)}$

$$\begin{aligned} 27_{(16)} &\Rightarrow 0010\ 0111_{(2)} \\ \text{CPLV } 27_{(16)} &\Rightarrow 1101\ 1000_{(2)} \\ \text{CPL2 } 27_{(16)} &\Rightarrow 1101\ 1001_{(2)} \\ \text{CPL2 } 27_{(16)} &= D9_{(16)} \end{aligned}$$

Remarque : on peut arriver plus rapidement à ce résultat en faisant :

$$100_{(16)} - 27_{(16)} = D9_{(16)} \text{ ; } (100_{(16)} = FF_{(16)} + 1)$$

* Pourquoi utiliser la notation avec complément ?

Une soustraction A - B peut s'écrire :

A - B = A + (R - B) modulo R.

Exemple en base 10 :

529 - 412 = 529 + (1000 - 412) modulo 1000 soit 117.
or 117 = (529 + 588) modulo 1000 = 1117 modulo 1000.

On observe que :

1/ Une soustraction se traite comme une addition si on fait intervenir le complément.
2/ Il est nécessaire de formater le complément. Le bit de poids le plus fort est le bit de signe. Par convention," 0" pour le signe "+", "1" pour le signe "-".

Exemples de calculs :

* Notation :

Exprimons le chiffre algébrique "-7" en binaire sur 4 bits :

+7 ⇔ 0 1 1 1
-7 ⇔ 1 1 0 0 1 (obtenu par complément à 2)

* Exemples de soustractions :

1er cas : soustraction avec résultat positif sur quatre bits.

Représentons l'opération : 7 - 2 = 5

+7..⇨....0 1 1 1
(+2 ⇨ 0 0 1 0)
-2..⇨..1 1 1 1 0

....0 0 1 0 1

1 + 0 = 1
1 + 1 = 0 et je retiens 1
1 + 1 + 1 (retenue) = 1 et je retiens 1
1 + 1 (retenue) = 0 et je retiens 1
1 + 1 = 0 Le bit de rang 5 est le bit qui définit le signe de la valeur trouvée. Ici la valeur est positive. Le bit de rang 6 n'a pas d'intérêt pour notre opération.

Résultat : 0 0 1 0 1 ⇨ +5

2ième cas : résultat négatif sur 5 bits.

Représentons l'opération : 3 - 8 = - 5

+3..⇨....0 0 1 1
(+8 ⇔ 1 0 0 0)
-8..⇨..1 1 0 0 0

1 1 0 1 1

1 + 0 = 1
1 + 0 = 1
0 + 0 = 0
0 + 1 = 1
0 + 1 = 1

Résultat :

1 1 0 1 1 résultat complémenté puisque le MSB = 1

CPLV 1 1 0 1 1 = 0 0 1 0 0

CPL2 1 1 0 1 1 = 0 0 1 0 1 ⇨ 5

Le résultat est donc, en tenant compte du MSB =1 de -5...

Opérations sur mots logiques :

Souvent en automatisme on est amené à travailler par comparaison sur des états de variables internes (bits internes, états d'étapes,…). Les opérations sur mots logiques vont permettre de réaliser ce travail par masquage de certaines variables. On peut utiliser ce principe pour synchroniser un automate asynchrone.

ET logique sur 8 bits :

```
0 1 0 0 0 1 1 0
                |   AND
0 1 1 1 0 1 0 0 |
----------------
0 1 0 0 0 1 0 0  ← résultat logique
```

Ce qui fait en base 16 : $46_{(16)}$ AND $74_{(16)}$ = $44_{(16)}$

Ce qui fait en base 10 : $70_{(10)}$ AND $116_{(10)}$ = $68_{(10)}$

OU logique sur 8 bits :

```
0 1 0 0 0 1 1 0
                |   OR
0 1 1 1 0 1 0 0 |
----------------
0 1 1 1 0 1 1 0  ← résultat logique
```

Ce qui fait en base 16 : $46_{(16)}$ OR $74_{(16)}$ = $76_{(16)}$

Ce qui fait en base 10 : $70_{(10)}$ OR $116_{(10)}$ = $118_{(10)}$

OU EXCLUSIF logique sur 8 bits :

```
0 1 0 0 0 1 1 0
                |   XOR
0 1 1 1 0 1 0 0 |
----------------
0 0 1 1 0 0 1 0  ← résultat logique
```

Ce qui fait en base 16 : $46_{(16)}$ OR $74_{(16)}$ = $32_{(16)}$

Ce qui fait en base 10 : $70_{(10)}$ OR $116_{(10)}$ = $50_{(10)}$

Le code ASCII

Le code ASCII (American Standard Code for Information Interchange) est le codage qui est utilisé en informatique pour communiquer entre le clavier d'un micro-ordinateur et l'unité centrale. Il y

a deux codes ASCII, le code ASCII standard et le code ASCII étendu. Le clavier est équipé d'un microprocesseur du type Intel 8048 qui scrute les circuits du clavier en permanence. Chaque touche possède un code distinct. Le code ASCII standard possède 127 caractères. Le code ASCII étendu possède 255 caractères. Il faudra donc pour coder l'ensemble des caractères utiliser 7 bits (du bit 0 au bit 6) pour le code ASCII standard et 8 bits pour le code ASCII étendu (du bit 0 au bit 7) . Le code ASCII différencie les lettres majuscules des lettres minuscules. Par exemple, pour écrire "A", le microprocesseur du clavier envoie à l'unité centrale le code $41_{(16)}$, pour écrire "a", le microprocesseur envoie à l'unité centrale le code $61_{(16)}$. L'espace entre deux caractères c'est $20_{(16)}$.
Remarque : b6 est le MSB et b0 est le LSB.

Tableau du code ASCII :

b6	0	0	0	0	1	1	1	1
b5	0	0	1	1	0	0	1	1
b4	0	1	0	1	0	1	0	1

b3	b2	b1	b0								
0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	~	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	¦
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

"q" ⇔ $111\ 0001_{(2)}$
"q" ⇔ $113_{(10)}$
"q" ⇔ $71_{(16)}$

"?" ⇔ $011\ 1111_{(2)}$
"?" ⇔ $63_{(10)}$
"?" ⇔ $3F_{(16)}$

Déroulement du TP

Remarque : ce TP n'est pas important, il est très important.

Après avoir lu le support traitez les questions suivantes :

1 - Ecrire en base 2 les nombres suivants :

56; 115; 152; 524; 615; 1020.

2 - Ecrire en base 16 les nombres suivants :

56; 115; 152; puis, 524; 615; 1020.

3 - Ecrire les nombres hexadécimaux en décimal, puis en binaire :

A6F; 128; 3AD; FFF; FAB; EC7; 100; DDD.

4 - Classer dans l'ordre croissant les nombres suivants :

$11111001_{(2)}$; $1101_{(10)}$; $1101_{(16)}$; $1000_{(16)}$; $1000_{(2)}$; $10000_{(10)}$

5 - Ecrire en hexadécimal le complément vrai du mot $F8_{(16)}$.

6 - Ecrire en hexadécimal le complément à 2 du mot $35_{(16)}$.

7 - Effectuer en binaire l'opération 9 - 6 = 3.

8 - Effectuer en binaire l'opération 6 - 9 = -3.

9 - Ecrire les codes donnés par le clavier à l'unité centrale pour inscrire la phrase suivante sur l'écran :

J'aime l'A.I.I...

Eléments de corrigés

1 - Ecrire en base 2 les nombres suivants :

56 ⇨ $111000_{(2)}$	115 ⇨ $1110011_{(2)}$	152 ⇨ $10011000_{(2)}$
524 ⇨ $1000001100_{(2)}$	615 ⇨ $1001100111_{(2)}$	1020 ⇨ $1111111100_{(2)}$

2 - Ecrire en base 16 les nombres suivants :

56 ⇨ $38_{(16)}$	115 ⇨ $73_{(16)}$	152 ⇨ $98_{(16)}$
524 ⇨ $20C_{(16)}$	615 ⇨ $267_{(16)}$	1020 ⇨ $3FC_{(16)}$

3 - Ecrire les nombres hexadécimaux en décimal, puis en binaire :

A6F ⇨ 2671 $_{(10)}$	128 ⇨ 296 $_{(10)}$
A6F ⇨ 1010 0110 1111 $_{(2)}$	128 ⇨ 1 0010 1000 $_{(2)}$

FAB ⇨ 4011 $_{(10)}$	EC7 ⇨ 3783 $_{(10)}$
FAB ⇨ 1111 1010 1011 $_{(2)}$	EC7 ⇨ 111011000111 $_{(2)}$

3AD ⇨ 941 $_{(10)}$	FFF ⇨ 4095 $_{(10)}$
3AD ⇨ 11 1010 1101 $_{(2)}$	FFF ⇨ 1111 1111 1111 $_{(2)}$

100 ⇨ 256 $_{(10)}$	DDD ⇨ 3549 $_{(10)}$
100 ⇨ 1 0000 0000 $_{(2)}$	DDD ⇨ 110111011101 $_{(2)}$

.

4 - Classer dans l'ordre croissant les nombres suivants :

$11111001_{(2)}$ ⇨ 249 $_{(10)}$		1101 $_{(10)}$ ⇨ 1101 $_{(10)}$	
$1101_{(16)}$ ⇨ 4353 $_{(10)}$		1000 $_{(16)}$ ⇨ 4096 $_{(10)}$	
$1000_{(2)}$ ⇨ 8 $_{(10)}$		10000 $_{(10)}$ ⇨ 10000 $_{(10)}$	

$1000_{(2)} < 11111001_{(2)} < 1101_{(10)} < 1000_{(16)} < 1101_{(16)} < 10000_{(10)}$

5 - Ecrire en hexadécimal le complément vrai de $F8_{(16)}$:

FF - F8 = 7

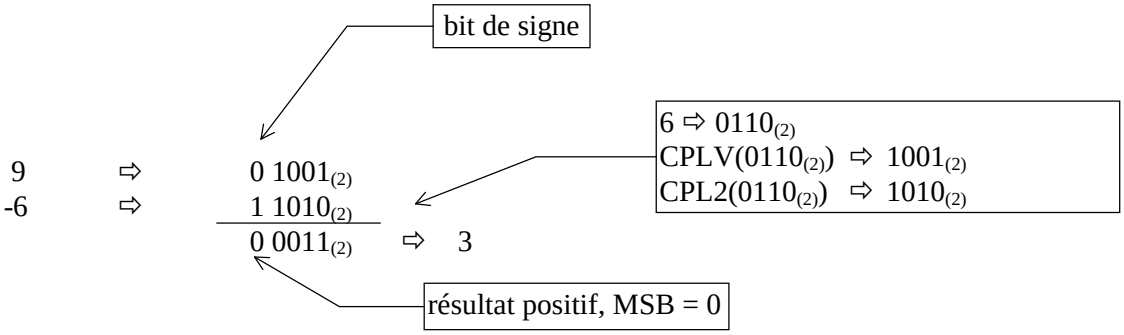
6 - Ecrire en hexadécimal le complément à 2 de $35_{(16)}$:

Le complément de 35 $_{(16)}$ est : FF - 35 = <u>CA</u>

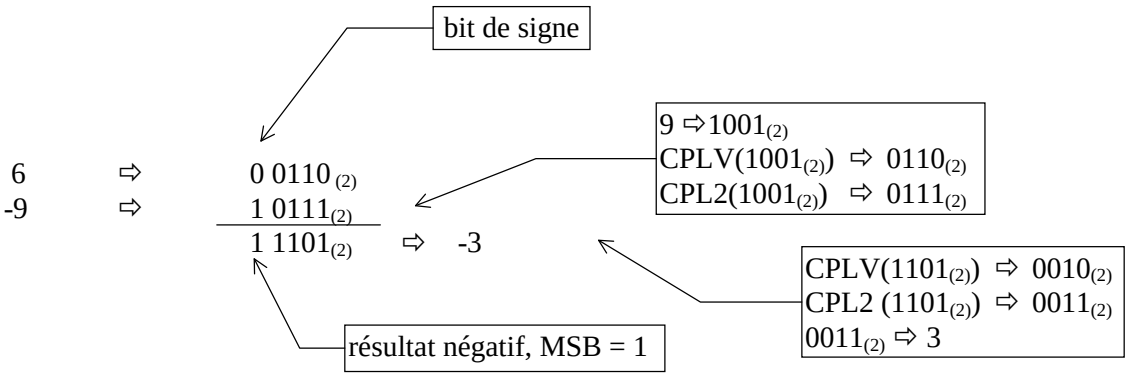
Le complément à 2 de $35_{(16)}$ est : CA + 1 = CB

CPL2 (35) = CB

7 - Effectuer l'opération 9 - 6 = 3.(Calcul effectué sur 4 bits, le 5ème bit étant le bit de signe)



8 - Effectuer l'opération 6 - 9 = -3.



9 - Ecrire les codes hexadécimaux que doit envoyer le clavier à l'unité centrale pour inscrire la phrase suivante sur l'écran :

J'aime l'A.I.I...

4A 27 61 69 6D 65 20 6C 27 41 2E 49 2E 49 2E 2E 2E