IUT AISNE	La numération	Semestre 1
BUT GMP	Cours	I.S.C.

1) Présentation du binaire

Vers la fin des années 30, Claude Shannon démontra qu'à l'aide de « contacteurs » (interrupteurs) fermés pour « vrai » et ouverts pour « faux » il était possible d'effectuer des opérations logiques en associant le nombre 1 pour « vrai » et 0 pour « faux ».

Ce codage de l'information est nommé **base binaire**. C'est avec ce codage que fonctionnent les ordinateurs. Il consiste à utiliser deux états (représentés par les chiffres 0 et 1) pour coder les informations.
L'homme calcule depuis 2000 ans avant Jésus-Christ avec 10 chiffres (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), on parle alors de base décimale (ou base 10). Toutefois dans des civilisations plus anciennes ou pour certaines applications actuelles d'autres bases de calcul ont et sont toujours utilisées :

- base sexagésimale (60), utilisée par les Sumériens. Cette base est également utilisée dans le système horaire actuel, pour les minutes et les secondes ;

- base vicésimale (20), utilisée par les Mayas ;

- base duodécimale (12), utilisée par les anglo-saxons dans leur système monétaire jusqu'en 1960 : un « pound » représentait vingt « shilling » et un « shilling » représentait douze « pences ». Le système d'heure actuel fonctionne également sur douze heures (notamment dans la notation anglo-saxonne) ;

- base quinaire (5), utilisée par les Mayas ;

- base binaire (2), utilisée par l'ensemble des technologies numériques.

2) Représentation des nombres en binaire

→ L'homme actuel est habitué dans la vie courante à manipuler des valeurs numériques en base 10

Exemple :

Lorsque nous écrivons le nombre 528 dans le système décimal, chacun des chiffres a un rang, encore appelé poids déterminé. Un chiffre est aussi appelé digit.

Le chiffre **8** représente **8 unités** soit :
Le chiffre **2** représente **2 dizaines** soit :
Le chiffre **5** représente **5 centaines** soit :

Le système décimal est un système de base ; il faut donc 10 caractères qui sont les chiffres de à .

→ les microprocesseurs et les ordinateurs traitent les valeurs numériques en binaire.

Le **système binaire** étant un système de **base 2**, il faut donc 2 caractères pour représenter les nombres **0** (faux) et **1** (vrai).

Chaque terme d'un nombre binaire est appelé **bit** (binary digit). Avec un bit, il est possible d'obtenir quatre états différents (2×2)

Exemple de nombre binaire : **1 0 0 1 1 0 1**

Le bit le plus à droite d'un nombre binaire est appelé (Least Significant Bit), c'est le **bit de poids le plus .**

Le bit le plus à gauche est appelé (Most Significant Bit), c'est le bit de poids le plus .

Dans le système binaire, le **kilo vaut** 2^{10} = , donc 2k bits = bits.

Il faut retenir qu'un **mot de 8 bits** est appelé ; en anglais **OCTET** se traduit par " ".

Exemple : certains ordinateurs, consoles de jeux, etc. ..., "travaillent" sur 16 bits, 32 bits ou 64 bits soit 16, 32 ou 64 informations binaires envoyées en même temps. Plus il y a d'informations envoyées a des intervalles de temps très courts, plus le système travaillera vite.

Remarque : Vous achetez un disque dur externe de 1To, vous le branchez sur votre ordinateur et celui-ci vous renseigne d'une capacité de 931 Go. D'où vient la différence ?

KiloOctets, MégaOctets

Longtemps l'informatique s'est singularisée par l'utilisation de différentes valeurs pour les unités du système international. Ainsi beaucoup d'informaticiens ont appris que 1 kilooctet valait 1024 octets. Or, depuis décembre 1998, l'organisme international "***IEC***" a statué sur la question voici donc les unités standardisées :

- Un kilooctet (ko ou kB) = 1000 octets
- Un Mégaoctet (Mo ou MB) = 1000 ko = 1 000 000 octets
- Un Gigaoctet (Go ou GB) = 1000 Mo = 1 000 000 000 octets
- Un Téraoctet (To) = 1000 Go = 1 000 000 000 000 octets

Attention ! De nombreux logiciels (parfois même certains systèmes d'exploitation) utilisent toujours la notation antérieure à 1998 pour laquelle :

Un kilooctet (ko) = 2^{10} octets = 1024 octets
Un Mégaoctet (Mo) = 2^{20} octets = 1024 ko = 1 048 576 octets
Un Gigaoctet (Go) = 2^{30} octets = 1024 Mo = 1 073 741 824 octets
Un Téraoctet (To) = 2^{40} octets = 1024 Go = 1 099 511 627 776 octets

L'IEC a également défini le kilo binaire (kibi), le méga binaire (Mébi), le giga binaire (Gibi), le téra binaire (Tebi).
Voici leurs définitions :

- Un kibioctet (kio ou kiB) vaut 2^{10} = 1024 octets
- Un Mébioctet (Mio ou MiB) vaut 2^{20} = 1 048 576 octets
- Un Gibioctet (Gio ou GiB) vaut 2^{30} = 1 073 741 824 octets
- Un Tébioctet (Tio ou TiB) vaut 2^{40} = 1 099 511 627 776 octets

Il est également utile de noter que la communauté internationale dans son ensemble utilise préférentiellement le nom de « byte » plutôt que le terme « octet » purement francophone. Cela donne les notations suivantes pour kilobyte, mégabyte, gigabyte et terabyte : *kB, MB, GB, TB*

Notez l'utilisation d'un *B* majuscule pour différencier *Byte* et *bit*.

Exercice 1 :
Un ordinateur dispose d'un disque "dur" (mémoire permanente de l'ordinateur) de 40 Mo (Mégaoctet) ; combien d'informations binaires (bits) peut-on stocker dans ce disque dur ?

Exercice 2 :
Un ordinateur dispose d'un disque "dur" (mémoire permanente de l'ordinateur) de 2 GiB (Gibi Byte) ; combien d'informations binaires (bits) peut-on stocker dans ce disque dur ?

3) CONVERSIONS

3.1) Conversion binaire - décimal

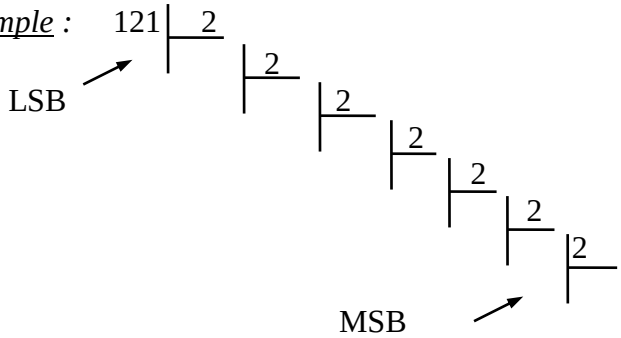
Méthode : nous utiliserons les puissances successives de deux.

Exemple : 1101 →
110,10 →

3.2) Conversion décimal - binaire

Méthode : nous utiliserons le méthode des divisions entières successives.

Exemple :



En reprenant les restent du dernier au premier, nous trouvons : **121 →**

Exercices : 322, 15 et 517 à convertir en binaire

3.3) Tableau de conversion décimal - binaire

Tableau de conversion :				
Binaire				**Décimal**
2^3	**2^2**	**2^1**	**2^0**	
8	**4**	**2**	**1**	
0	**0**	**0**	**0**	
0	**0**	**0**	**1**	
0	**0**	**1**	**0**	
0	**0**	**1**	**1**	
0	**1**	**0**	**0**	
0	**1**	**0**	**1**	
0	**1**	**1**	**0**	
0	**1**	**1**	**1**	
1	**0**	**0**	**0**	
1	**0**	**0**	**1**	
1	**0**	**1**	**0**	
1	**0**	**1**	**1**	
1	**1**	**0**	**0**	
1	**1**	**0**	**1**	
1	**1**	**1**	**0**	
1	**1**	**1**	**1**	

4) OPERATIONS ARITHMETIQUES EN BINAIRE

Il ne faut pas perdre de vue que les calculs seront fait par des automatismes (machines électriques) ; ceci nous impose de retenir que la longueur des nombres, encore appelée format, est fixe.

Dans la suite, sauf indication contraire, le format sera de huit bits soit l'octet, format très souvent employé dans les automatismes électroniques.

4.1) L'addition

Les quatre additions de bases sont : 0 + 0 =
0 + 1 =
1 + 0 =
1 + 1 = avec une retenue de

Exemple :

```
  10111010
+  1101100
----------
100100110
```



En informatique cette retenue est appelée "**overflow**" qui signifie dépassement de capacité

Exercices :

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

4.2) La soustraction

Les quatre soustractions de base sont : 0 - 0 =
1 - 0 =
0 - 1 = avec une retenue de
1 - 1 =

La soustraction est légèrement plus difficile à maîtriser pour un circuit électronique.

Nous verrons plus loin une méthode permettant d'effectuer plus simplement des soustractions par la méthode du complément à deux.

Exemple :

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

Exercices :

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

4.3) La multiplication

Les quatre multiplications de base sont : 0 × 0 =
1 × 0 =
0 × 1 =
1 × 1 =

Exemple :

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

Exercices :

1 1 0 1	1 0 1	1 0 1 0
× 1 1 1	× 1 0 1	× 1 0

4.4) La division

La division est une suite de soustraction et de comparaison. Elle est très peu employée en électronique donc inutile de la développer.

5) Le système hexadécimal

L'**hexadécimal** est un système de **base** , il est donc composé de caractères différents. Les chiffres utilisés sont ceux de la base 10 de à auxquels viennent s'ajouter les lettres à .

Exemple de nombre de base 16 : 2 E 5 ; 5 6 7 ; C A F E

Nous remarquerons qu'il sera souvent nécessaire de préciser la base d'un nombre, celle - ci sera mise en indice. *Exemple :* 125_{10} ; 255_{16} .

D'autre part, l'utilisation des indices permet d'utiliser le signe égal (=) entre deux nombres de bases différentes. *Exemple* : $1\ 0\ 1_2 = 5_{10}$

5.1) Conversion hexadécimal en binaire et BIN - HEXA

Méthode : les conversions se feront chiffre à chiffre par mot de 4 bits à l'aide du tableau de conversion ci-dessous :

Tableau de conversion :

Décimal	Binaire				Hexadécimal
	2^3	2^2	2^1	2^0	
	8	4	2	1	
	0	0	0	0	
	0	0	0	1	
	0	0	1	0	
	0	0	1	1	
	0	1	0	0	
	0	1	0	1	
	0	1	1	0	
	0	1	1	1	
	1	0	0	0	
	1	0	0	1	
	1	0	1	0	
	1	0	1	1	
	1	1	0	0	
	1	1	0	1	
	1	1	1	0	
	1	1	1	1	

Exemples : 2 → ; E → ; F → ; 0 → ; 2 E → .

6) CODES BINAIRES

6.1) Le code binaire naturel

Le binaire naturel est un code qui permet les opérations arithmétiques comme nous l'avons vu précédemment. Ce code est aussi appelé code binaire pur direct.

6.2) Le code BCD ou décimal codé binaire

- Les systèmes digitaux utilisent tous des nombres binaires pour leurs opérations internes.
- La numérotation décimal étant la plus couramment utilisée par les humains, cela suppose une conversion entre la numérotation en décimal et la numérotation en binaire.

- Si chaque digit d'un nombre décimal est représenté par son équivalent binaire, cela constitue le code appelé **BCD**.

- Un digit décimal peut atteindre la valeur 9, il faut donc nécessairement 4 bits pour coder chaque digit (le code binaire de 9 étant 1 0 0 1).

- Ce code est identique au binaire naturel jusque 9 ; il utilise une représentation binaire des chiffres sur quatre bits.

Exemple : $8\,9_{10}$ = $_{BCD}$; $8\,7\,4_{10}$ = $_{BCD}$;
$1\,0\,0\,1\;\;0\,0\,1\,1_{BCD}$ = $_{10}$

Remarque : comparaison entre le code BCD et le code binaire naturel :

- il est important de constater qu'un nombre BCD n'est pas semblable à un nombre en binaire naturel au-delà de 9.

- Le code BCD converti chaque digit décimal individuellement en binaire.

6.3) Le code binaire réfléchi ou code GRAY

Le binaire réfléchi est un code qui lorsqu'on passe d'un terme au suivant .

Remarque : en binaire naturel, le passage d'un nombre au nombre suivant se traduit par un changement de un ou plusieurs bits. Le passage de 3 à 4 en décimal donne en binaire 0 1 1 à 1 0 0 → les trois bits changent simultanément.

Construction du code GRAY :

- Les deux premiers termes sont identiques à ceux du binaire naturel.

- Les deux termes suivants sont obtenus en faisant la symétrie de la première colonne et en inscrivant des 1 dans la deuxième colonne.

- Les quatre suivant par symétrie des deux premières colonnes et en mettant des 1 dans la troisième colonne.

- Les huit suivants par symétrie des trois premières colonnes et en mettant des 1 dans la quatrième colonne.

- etc.

Ce code est utilisé dans la représentation des tableaux de ***Karnaugh***.

Code binaire « Gray »			
0	**0**	**0**	**0**
0	**0**	**0**	**1**
0	**0**	**1**	**1**
0	**0**	**1**	**0**
0	**1**	**1**	**0**
0	**1**	**1**	**1**
0	**1**	**0**	**1**
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**

6.4) Le code complément à 1

Ce code est un code signé, c'est-à-dire qu'il permet la représentation de nombres négatifs.

Soit à effectuer l'opération D = A - B
Premier cas : |A| > |B|

Procédure :

a) Ecrire le complément à 1 de B, soit $\overline{B}$, revient à inverser tous les chiffres d'un nombre.

Dans $\overline{B}$ le bit situé le plus à gauche est appelé bit de signe.
Si le bit de signe est égal à :
- à 1, le nombre est négatif,
- à 0, le nombre est positif.

b) Effectuer l'opération X = A + $\overline{B}$ qui génère une retenue **R**.
Le resultat, **D**, de la soustraction est égal à :
X + R
Ce résultat étant forcément positif, le bit de signe n'est pas pris en compte.

Application numérique :

$A = 1\ 1\ 0\ 1_2$ soit 13_{10}
$B = 0\ 1\ 1\ 0_2$ soit 6_{10}

$\overline{B} = 1\ 0\ 0\ 1$
↖ bit de signe, ($\overline{B}$ est un nombre négatif)

		soit A
	+ 1 0 0 1	soit $\overline{B}$
Retenue ⌫		soit X
	+ **1**	
D =		soit

Deuxième cas : |A| ≤ |B|
Dans ce cas le résultat D de la soustraction est négatif, son signe est confirmé par examen du bit de signe.

Procédure :

a) Ecrire le complément à 1 de A, soit $\overline{A}$ et étudier son bit de signe.

b) Effectuer l'opération X = B + $\overline{A}$ qui ne génère aucune retenue.

Le résultat, D, de la soustraction est égal à :
X + R

Ce résultat devant être négatif, le bit de signe doit être vérifié.

Application numérique :

$A = 0\ 1\ 1\ 0$ soit 6_{10}
$B = 1\ 1\ 0\ 1$ soit 13_{10}

$\overline{A} = 1\ 0\ 0\ 1$
↖ Bit de signe, ($\overline{A}$ est un nombre négatif)

		soit B
	+ 1 0 0 1	soit $\overline{A}$
Retenue ⌫		soit X
	+ **1**	
D =		soit

↖ Bit de signe, (D est négatif)

6.5) Code complément à deux

Ce code permet également la représentation de nombres négatifs.

Les nombres positifs sont identiques au binaire naturel avec un bit supplémentaire à zéro placé à gauche du nombre. Les nombres négatifs sont constitués par les compléments à 2 (complément 1 + 1) des nombres positifs.

Pour complémenter à deux, il faut inverser tous les chiffres du nombre (complément à 1) et ajouter 1 (+1).

Soit à effectuer l'opération D = A - B

Premier cas : |A| > |B|

Procédure :

a) Ecrire le complément à 1 de B, soit $\overline{B}$

b) Faire $\overline{B}$ + 1 = -B.
Le résultat D de la soustraction est égal à :

A + (-B)

Ce résultat étant forcément positif, le bit de signe n'est pas pris en compte.

Application numérique :

A = 1 1 0 1_2 soit 13_{10}
B = 0 1 1 0_2 soit 6_{10}

$\overline{B}$ = 1 0 0 1

bit de signe, ($\overline{B}$ est un nombre négatif)

\+ 1
1 0 1 0

soit A
\+ 1 0 1 0 soit -B
D = soit

Deuxième cas : |A| ≤ |B|

Dans ce résultat, D de la soustraction est négatif,
son signe est confirmé par examen du bit de signe.

Procédure :

a) Ecrire le complément à 1 de A, soit $\overline{A}$

b) Faire $\overline{A}$ + 1 = -A.
Le résultat D de la soustraction est égal à :
B + (-A)

Ce résultat étant négatif, le bit de signe doit être vérifié.

Application numérique :

A = 0 1 1 0_2 soit 6_{10}
B = 1 1 0 1_2 soit 13_{10}

$\overline{A}$ = 1 0 0 1
bit de signe, ($\overline{A}$ est un nombre négatif) + 1

soit B
\+ 1 0 1 0 soit -A
D = soit

bit de signe, (D est négatif)

6.6) Code avec bit de parité

Ceux sont des codes qui permettent la détection d'erreur lors de la transmission d'informations binaires.

Il faut convenir entre le récepteur et l'émetteur d'un format ; On peut, par exemple, décider de l'état binaire "1" pour indiquer qu'un nombre est paire.

Exemple pour le binaire naturel : 0 →
1 →
2 →

Il existe un code permettant la correction d'erreur, c'est le code de HAMMING ; son défaut est de comporter 7 bits pour un codage de départ sur 4 bits.

6.7) Le code ASCII (Américan Standar Code for Information Interchange)

Comme son nom l'indique, c'est un code utilisé pour l'échange d'information du type alphanumérique. Il utilise 7 bits qui permettent le codage de 128 caractères (majuscules, minuscules, chiffres, divers symboles, divers commandes ...etc.). Il est très utilisé en informatique.

<u>Le tableau suivant représente la liste du code ASCII à 7 bits.</u>

CODE ASCII

VERSION INTERNATIONALE DE REFERENCE

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	↘	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC 1	!	1	A	Q	a	q
0	0	1	0	2	SXT	DC 2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC 3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC 4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

<u>Exemple de lecture</u> :

7 6 5 4 3 2 1
↓ ↓ ↓ ↓ ↓ ↓ ↓

La lettre U représentée par le groupe code →

La touche SP (espace ou blanc) est représentée par →