2. Représentation des données

- **Représentation dans l'ordinateur des :**
    - **Nombres entiers**
    - **Nombres décimaux**
    - **Caractères**
    - **Son**
    - **Images**
    - **Vidéos**

« Codage de l'information »

2. Représentation des données

Base de numération

- Base de numération = **manière de représenter un nombre**
- Un **nombre** s'écrit dans une base donnée B sous la forme **d'additions des puissances successives de cette base**
- **Le même nombre peut être représenté dans n'importe quelle base**
- Le choix de la base de représentation **dépend de la facilité à réaliser des calculs et de son utilité pour un but donné**

- On peut compter sur nos dix doigts…
  → **Base dix (ou décimale) : B = 10**
  → **utilise les chiffres de 0 à 9**

2. Représentation des données

Base décimale

- Utilise **10 chiffres** : de **0 à 9** (0 à 10-1)
- Ecriture d'un nombre en base 10 :
  avec des puissances successives de 10
- Lorsqu'on fait apparaître les puissances de 10, on dit qu'on « décompose » le nombre en base 10

- **Exemple de décomposition en base 10**

  4658 = 4*1000 + 6*100 + 5*10 + 8

  $4658 = 4*10^3 + 6*10^2 + 5*10^1 + 8*10^0$

- **Pour un nb. à *n+1* chiffres écrit en base 10 :** $\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_{10} = \sum_{k=0}^{n} a_k \times 10^k$

$$a_n a_{n-1} a_{n-2} \ldots a_1 a_0 = a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-1} \times 10^{n-2} + \ldots + a_1 \times 10 + a_0$$

Base B quelconque

- Dans une base B quelconque, on utilise **B chiffres** entre **0 et B-1**

- **Décomposition en base B : à l'aide des puissances de B :**

$$\boxed{a_n a_{n-1} a_{n-2} \ldots a_1 a_0 = a_n \times B^n + a_{n-1} \times B^{n-1} + a_{n-1} \times B^{n-2} + \ldots + a_1 \times B + a_0}$$

$$\boxed{\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_B = \sum_{k=0}^{n} a_k \times B^k}$$

- ***Cas particuliers :***

B = 10 → base décimale $\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_{10} = \sum_{k=0}^{n} a_k \times 10^k$

B = 2 → base binaire $\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_{2} = \sum_{k=0}^{n} a_k \times 2^k$

B = 8 → base octale $\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_{8} = \sum_{k=0}^{n} a_k \times 8^k$

2. Représentation des données

IUT Béziers

Représentation des données

- Dans la vie de tous les jours :
  - Différents formats qu'on peut lire, comprendre et manipuler (nombres en base 10, texte, sons, images…)
- A l'intérieur des ordinateurs :
  - Grandeurs physiques que l'ordinateur puisse manipuler
  - Deux états : un état « 0 » et un état « 1 »
  - Exemples : tension électrique 0 V / 5V, capacité chargée/déchargée etc.
  - Il s'agit donc d'une **représentation binaire** (en base 2) = **succession de « bits »**
- **Bit** = **Bi**nary digi**t** (chiffre binaire en anglais)
- Mot binaire (« Word ») = séquence de bits
- Octet (« **Byte** ») = mot binaire de **8 bits**

Base Binaire (B =2)

- On utilise **2 chiffres** : **0 et 1**
- **Décomposition en base 2 : à l'aide des puissances de 2 :**

$$\left(a_n a_{n-1} a_{n-2} \ldots a_1 a_0\right)_2 = a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-1} \times 2^{n-2} + \ldots + a_1 \times 2 + a_0$$

$$\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_2 = \sum_{k=0}^{n} a_k \times 2^k$$

Bases utilisées en informatique

B = 2 → base binaire $\left(a_n a_{n-1} a_{n-2} \dots a_0\right)_2 = \sum_{k=0}^{n} a_k \times 2^k \quad a_k \in [0,1]$

B = 8 → base octale $\left(a_n a_{n-1} a_{n-2} \dots a_0\right)_8 = \sum_{k=0}^{n} a_k \times 8^k \quad a_k \in [0,7]$

B = 16 → base hexadécimale

$$\left(a_n a_{n-1} a_{n-2} \dots a_0\right)_{16} = \sum_{k=0}^{n} a_k \times 16^k$$

$$a_k \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

2. Représentation des données

Les 16 premiers nombres

Décimal	Binaire	Hexadécimal	Octal
00	0000	0	00
01	0001	1	01
02	0010	2	02
03	0011	3	03
04	0100	4	04
05	0101	5	05
06	0110	6	06
07	0111	7	07
08	1000	8	10
09	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

IUT Béziers

Exemples

Base binaire

2^3 2^2 2^1 2^0

$$1\ 1\ 0\ 1_2 = 1 + 0*2^1 + 1*2^2 + 1*2^3 = 1 + 4 + 8 = 13_{10}$$

Base octale

8^2 8^1 8^0

$$2\ 3\ 5_8 = \overset{8^0}{5} + 3*8^1 + 2*8^2 = 5 + 3*8 + 2*64 = 157_{10}$$

Base hexadécimale

16^1 16^0

$$F\ 5_{16} = \overset{.16^0}{5} + F*16^1 = 5 + 15*16 = 245_{10}$$

2.1 Représentation des nombres entiers positifs (non signés, « unsigned »)

Changement de base

- ***Base B → base décimale :***
  avec l'expression donnée précédemment

$$\left(a_n a_{n-1} a_{n-2} \ldots a_0\right)_B = \sum_{k=0}^{n} a_k \times B^k$$

- ***Base décimale → base B :***
  en réalisant des divisions successives par B, jusqu'à l'obtention d'un quotient < B

Changement de base

Base décimale → base B

- ***Soit* X_{10} *un nombre* X *écrit en base 10 qu'on veut écrire en une autre base B***

$X_{10} = q_0 \times B + r_0$ *avec q_0 le quotient de la division X_{10}/B et r_0 son* ***reste***

$$q_0 = q_1 \times B + r_1$$

$$q_1 = q_2 \times B + r_2$$

$$\ldots$$

$$q_{m-2} = q_{m-1} \times B + r_{m-1}$$

$$q_{m-1} = 0 \times B + r_m$$

$$\rightarrow \boxed{X_{10} = (r_m r_{m-1} \ldots r_2 r_1 r_0)_B}$$

Changement de base : exemple

On veut écrire le nombre 2838_{10} en octal

2838 / 8 = 354,75 soit 354 et un reste 0,75*8 = 6

354 / 8 =44,25 soit 44 et un reste 0,25*8 = 2

44 / 8 = 5,5 soit 5 et un reste 0,5 * 8 = 4

5 < 8 : c'est fini !

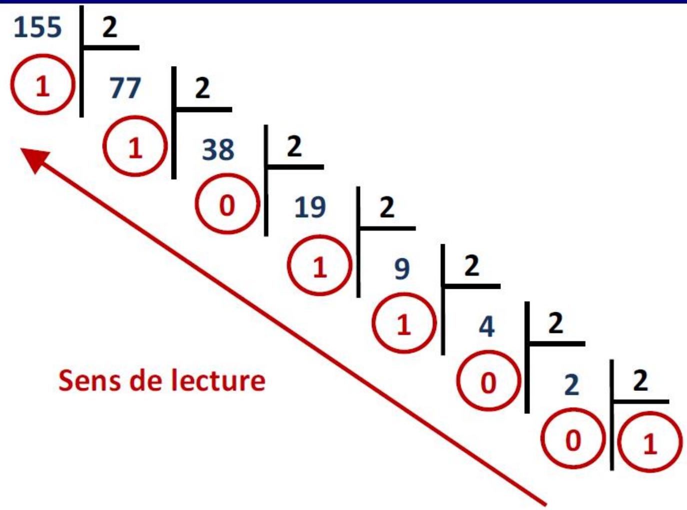
$2838_{10} = 5426_8$

2. Représentation des données

Passage de la base 10 à la base 2

- ***On peut utiliser les divisions successives par 2…***

Exemple



$155 = (10011011)_2$

2. Représentation des données

Passage de la base 10 à la base 2

… mais c'est plus simple si on connaît les puissances de 2

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}
1	2	4	8	16	32	64	128	256	512	1024	2048	4096

- ***Exemple : écrire 2838_{10} en binaire***

2838 = 2048 + 790

2838 = 2048 + 512 + 278

2838 = 2048 + 512 + 256 + 22

2838 = 2048 + 512 + 256 + 16 + 6

$2838_{10} = 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0_2$

Passage d'une base B à une base B^k

- *On regroupe les chiffres k par k*
- *On transforme chaque groupe de chiffres en base B^k*

- ***Exemple :** passage de **binaire en héxa** (B =2, k=4)*

101100010101_2 = **1011** 0001 **0101**$_2$ = **B 1 5**$_{16}$

***A retenir : 4 bits = 1 chiffre hexa** ($16 = 2^4$)*

Opérations élémentaires en binaire

- ***Addition***

0 + 0 = 0

0 + 1 = 1

1 + 1 = 10

- ***Multiplication***

0 * 0 = 0

0 * 1 = 0

1 * 1 = 1

2. Représentation des données

Exemples d'opérations en binaire

- ***Addition***

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

- ***Multiplication***

		1	0	1	1
*				1	0
		0	0	0	0
	1	0	1	1	
	1	0	1	1	0

Représentation des nombres dans les mémoires des ordinateurs

- ***Les nombres sont représentées dans les mémoires par des bits regroupés dans des octets ou des groupes d'octets***
  = entités de N bits (N = multiple de 8)

1 octet = 8 bits (N = 8)

Rq: 1 octet = 2 chiffres hexa

2 octets = 16 bits (N = 16)

4 octets = 32 bits (N = 32)

8 octets = 64 bits (N = 64)

Plage de valeurs et nombre de bits

- ***Les représentations internes de nombres sont de longueurs finies (et fixée) →***
  on ne peut pas représenter tous les nombres

- ***Une cellule (« mot ») de n bits permet de représenter 2^n valeurs différentes :***
  par exemple, des nombres positifs de 0 à 2^n-1

Exemple : ***sur 1 octet (8 bits), on peut représenter 2^8 valeurs différentes : entre*** **00000000** ***et*** **11111111**
soit des nombres entiers positifs entre **0** ***et*** **255**

(2^8-1 = 256-1 =255)

Débordement ou dépassement (« Overflow »)

- ***Que se passe-t-il si on demande à l'ordi de réaliser un calcul dont le résultat dépasse 2^n-1 (donc qui ne peut pas être représenté sur n bits) ?***

Exemple sur 8 bits :** on veut effectuer la somme **255 + 1

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

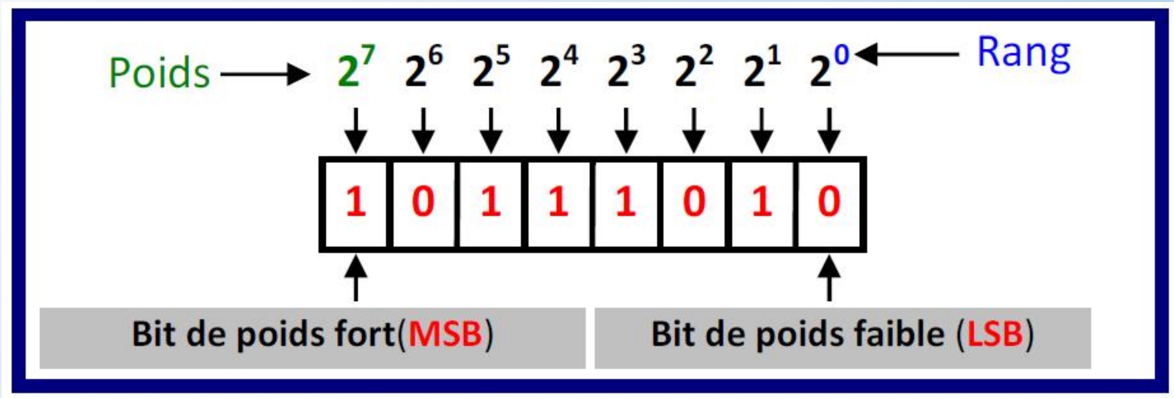
Le processeur donnera pour résultat 0 !

… car il ne dispose que de 8 bits…

MAIS il indiquera un dépassement (« overflow ») → Overflow Flag = 1 (OF =1)

Bit (octet) de poids fort,
Bit (octet) de poids faible

- ***Représentation d'un octet dans une mémoire***



- ***Pour les nombres représentés sur plusieurs octets, on parle, de la même manière, d'octet de poids fort et d'octet de poids faible***

Big endian, little endian

- ***Les nombres de plusieurs octets peuvent être représentés en mémoire***
  avec l'octet de poids fort en premier…

Exemple : le nombre hexa sur deux octets **A0 B7**
(noté 0*x*A0B7) sera représenté A0 B7
→ représentation « big endian »
Linux sur PA-RISC, SPARC, TCP/IP…

- ***… mais aussi avec l'octet de poids faible en premier :***

→ représentation « little endian » :
A0 B7 sera représenté B7 A0
(Windows, Linux sur x86…)

Big endian, little endian

- *« Endianess » = ordre dans lequel ces octets sont organisés en mémoire*
- *Il faut savoir quel « endianess » utilise la machine si l'on souhaite lire correctement les données brutes enregistrées, par exemple, sur un disque dur*

- *Exemple : listing des octets entre les adresses 200 et 21F d'un disque dur de PC*

```
0000200 e852 0128 0874 be56 8133 4ce8 5e01 f4bf
0000210 6681 2d8b 7d83 0008 840f 00e9 7c80 00ff
```

2. Représentation des données

- ***Points essentiels***
    - Savoir représenter un nombre en base 10, 2, 16
    - Savoir passer rapidement de la base 10 aux bases 2 et 16 et réciproquement
    - Savoir passer rapidement de la base 2 à la base 16 et réciproquement (octets, groupe d’octets)
    - Savoir effectuer des additions et des multiplications en base 2
    - Connaître la source d’un dépassement et son effet
    - Connaître les deux représentations des nombres entiers en hexa (« big » et « little » endian) et savoir reconstituer les nombres à partir de ces représentations

2. Représentation des données

- **Les représentations des :**

  - **Nombres entiers positifs**
  - **Nombres entiers avec signe**
  - **Nombres décimaux**
  - **Caractères**
  - **Son**
  - **Images**
  - **Vidéos…**

… seront traitée lors des dernières séances du cours

Et là-dedans, il y a quoi exactement ? ….