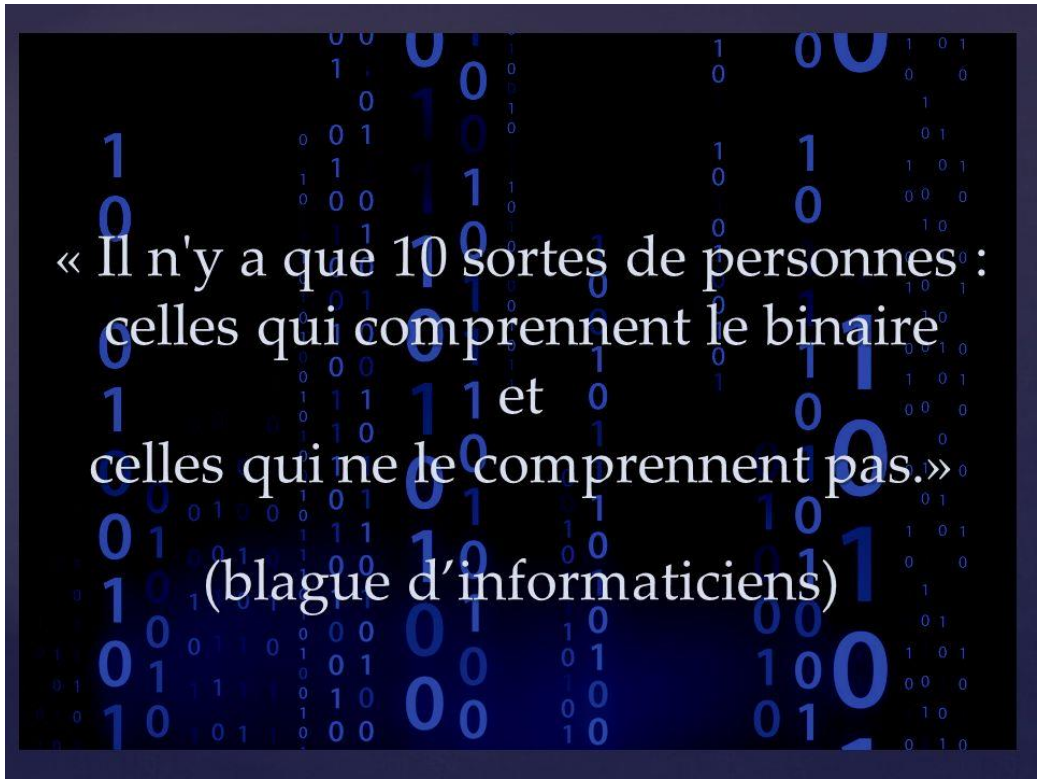
# INFORMATIQUE 1

## II. REPRESENTATION DES NOMBRES

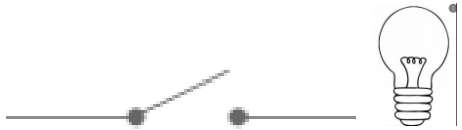



Eva Ansermin & Romuald Grignon

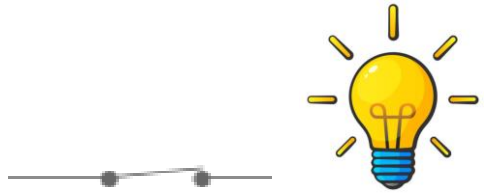
CY TECH

# Représentation des données

- Un ordinateur peut traiter/produire plusieurs types de données :
  - textes, chiffres
  - images
  - sons
  - …
- Toutes ces informations sont stockées et traitées sous forme **binaire** : avec des 0 et des 1.
- Cette unité d'information (0 0u 1) est le **binary digit** (**le bit**).
- Ce type de données est plus facile à gérer au niveau électronique : 0 correspond à un circuit ouvert, 1 à un circuit fermé.



0 : le courant ne passe pas



1 : le courant passe



Eva ANSERMIN & Romuald GRIGNON

# Représentation des données

- Un ordinateur peut traiter/produire plusieurs types de données :
  - textes, chiffres
  - images
  - sons
  - ...
- Toutes ces informations sont stockées et traitées sous forme **binaire** : avec des 0 et des 1.
- Cette unité d'information (0 ou 1) est le **binary digit** (**le bit**).

- **Comment sont représentés les nombres en binaire ?**



Eva ANSERMIN & Romuald GRIGNON

# Vocabulaire

- **Bit** : information élémentaire, valant 0 ou 1.
- **Octet** (Byte en Anglais) : information constituée de 8 bits.
- **Mot** : unité manipulée par le processeur (16, 32, 64bits, …).
- **Capacités** (norme commerciale depuis 1998).

| Nom | Capacité | Exemples |
|---|---|---|
| 1 octet | 8 bits | 1 octet ~ 2 caractères |
| 1 Ko (kilo octet) | $10^3$octets | 500 ko ~ un roman |
| 1 Mo (mega octet) | $10^6$ octets | 100 Mo ~ une pile de livre de 1 mètre |
| 1 Go (giga octet) | $10^9$ octets | 1 Go ~ une camionnette de livres |
| 1 To (téra octet) | $10^{12}$ octets | 2 To ~ une Bibliothèque Universitaire |
| 1 Po (peta octet) | $10^{15}$ octet | 2 Po ~ toutes les BU des USA |
| 1 Eo (exa octet) | $10^{18}$ octet | 5 Eo ~ tous les mots prononcés depuis le début de l'humanité<br>2 Eo ~ volume annuel des informations du monde |


Eva ANSERMIN & Romuald GRIGNON

# Vocabulaire

- **Bit** : information élémentaire, valant 0 ou 1.
- **Octet** (Byte en Anglais) : information constituée de 8 bits.
- **Mot** : unité manipulée par le processeur (16, 32, 64bits, …).
- **Capacités** (norme commerciale depuis 1998).

| Nom | Capacité | Capacité |
|---|---|---|
| 1 Kio (kibi octet) | $2^{10}$ | 1 024 octets |
| 1 Mio (mébi octet) | $2^{20}$ | 1 048 576 o |
| 1 Gio (Gibi octet) | $2^{30}$ | 1 073 741 824 o |
| 1 Tio (Tébi octet) | $2^{40}$ | 1 099 511 627 776 o |
| 1 Pio (Pébi octet) | $2^{50}$ | 1 125 899 906 842 620 o |
| 1 Eio (Exbi octet) | $2^{60}$ | 1 152 921 504 606 846 976 o |


Eva ANSERMIN & Romuald GRIGNON

# Les bases

- Nous comptons en base 10 (base décimale) : Les chiffres vont de 0 à 9.
- Exemple de décomposition d'un nombre :

$$542 = 5 * 100 + 4 * 10 + 2 * 1 = 5 * 10^2 + 4 * 10^1 + 2 * 1$$

- Soit N un entier dont les chiffres sont $\{a_n, a_{n-1}, a_{n-2}, \dots, a_0\}$ en base décimale:

$$N = a_n * 10^n + a_{n-1} * 10^{n-1} + \dots + a_0 * 10^0$$

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

# Les bases

- Nous comptons en base 10 (base décimale) : Les chiffres vont de 0 à 9.
- Exemple de décomposition d'un nombre :

$$542 = 5 * 100 + 4 * 10 + 2 * 1 = 5 * 10^2 + 4 * 10^1 + 2 * 1$$

- Soit N un entier dont les chiffres sont $\{a_n, a_{n-1}, a_{n-2}, \dots, a_0\}$ en base décimale:

$$N = a_n * 10^n + a_{n-1} * 10^{n-1} + \dots + a_0 * 10^0$$

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

i est le **poids** du chiffre.

On parle de **poids fort** pour le chiffre le plus à gauche (plus haute puissance) et **de poids faible** pour celui le plus à droite (puissance la plus faible)

# Les bases

- Nous utilisons plusieurs bases dans notre vie quotidienne:

| | |
|---|---|
| Base 1 | Comptage avec les doigts, cailloux |
| Base 2 | Système binaire, logique symbolique, *ordinateur* |
| Base 5 | Système quinaire, *Aztèques* |
| Base 7 | Notes musicales, jours de la semaine |
| Base 8 | Système octal, *ordinateur* |
| Base 10 | Système décimal, adopté aujourd'hui par l'homme |
| Base 12 | Monnaie et mesure anglaise, mois de l'année |
| Base 16 | Système hexadécimal, *ordinateur* |
| Base 20 | Comptage sur les doigts des mains et pieds, *Mayas* |
| Base 24 | Heures du jour |
| Base 60 | Degrés, minutes et secondes, *Babyloniens* |

# Les bases

- En base décimale : chiffre allant de 0 à 9 et $N = \sum_{i=0}^{n} a_i * 10^i$

- En **base binaire** (ou base 2) :
  - Les chiffres sont :

# Les bases

- En base décimale : chiffre allant de 0 à 9 et $N = \sum_{i=0}^{n} a_i * 10^i$

- En **base binaire** (ou base 2) :
  - Les chiffres sont : {0,1}

# Les bases

- En base décimale : chiffre allant de 0 à 9 et $N = \sum_{i=0}^{n} a_i * 10^i$

- En **base binaire** (ou base 2) :
  - Les chiffres sont : {0,1}
  - On multiplie les chiffres par des puissances de 2 :

  Soit N un entier dont les chiffres sont $\{p_n, p_{n-1}, p_{n-2}, \dots, p_0\}$ en base binaire :

$$N = \sum_{i=0}^{n} p_i * 2^i$$

$$N = p_n * 2^n + p_{n-1} * 2^{n-1} + \dots + p_0 * 2^0$$

# Les bases

- En base décimale : chiffre allant de 0 à 9 et $N = \sum_{i=0}^{n} a_i * 10^i$

- En **base binaire** (ou base 2) :
  - Les chiffres sont : {0,1}
  - On multiplie les chiffres par des puissances de 2 :

  Soit N un entier dont les chiffres sont $\{p_n, p_{n-1}, p_{n-2}, \dots, p_0\}$ en base binaire :

$$N = \sum_{i=0}^{n} p_i * 2^i$$

$$N = p_n * 2^n + p_{n-1} * 2^{n-1} + \dots + p_0 * 2^0$$

- Exemple : $011_2 = 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 0 + 2 + 1 = 3$

# Binaire -> décimal

- Il suffit d'ajouter les puissances de deux des chiffres dont le bit est à 1.

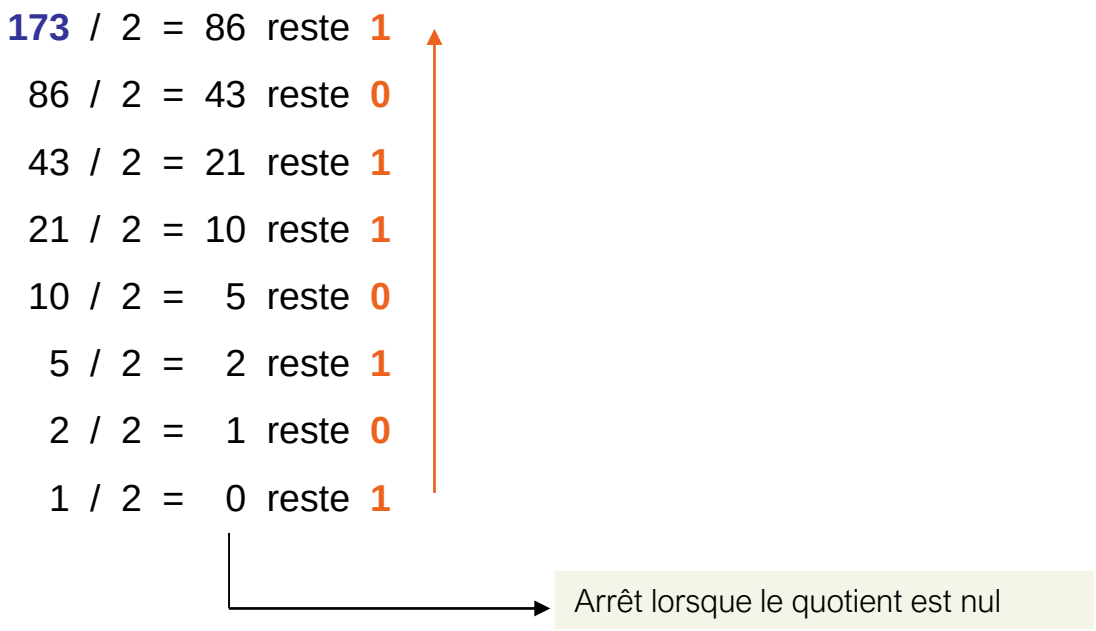
- Il faut donc connaître les puissances de deux !

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| | | | | | 1 | 0 | 0 | 1 | 0 | 0 |

- Exemple : $100100_2 \rightarrow 2^5 + 2^2 = 32 + 4 = 36$

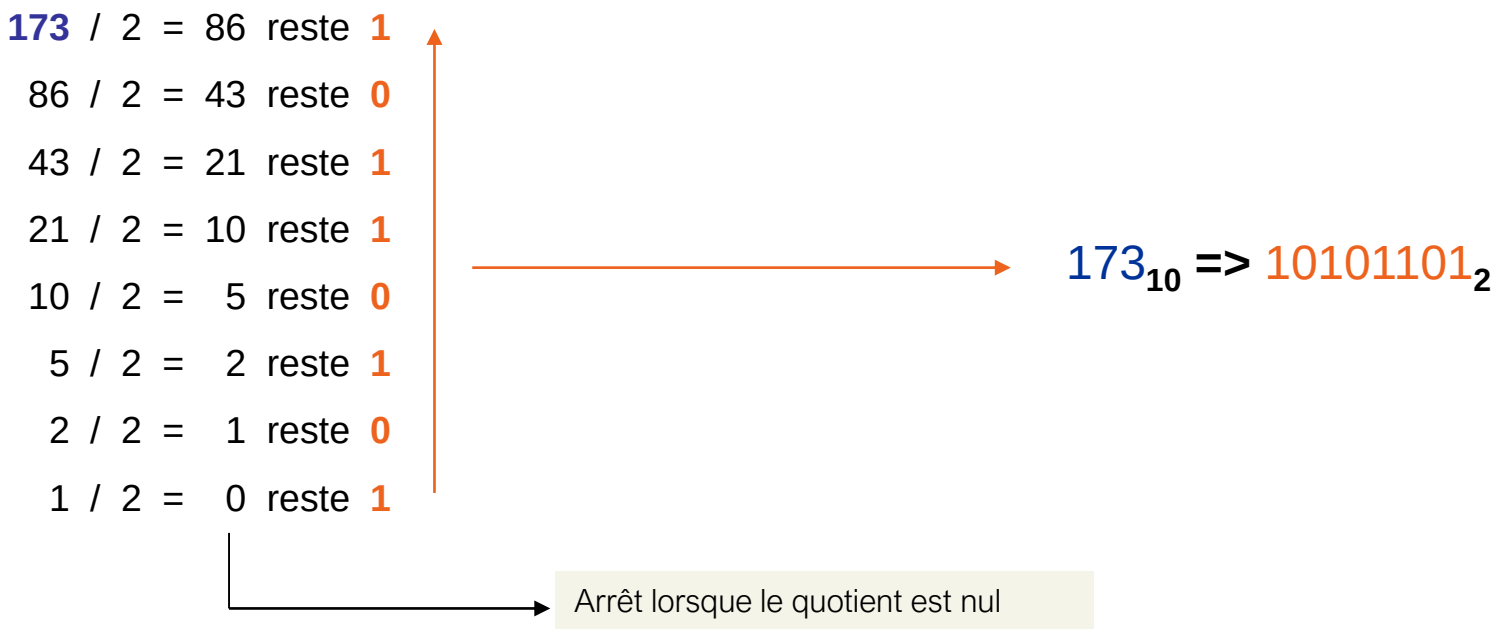
# Décimal -> binaire

- 1ère méthode : la division Euclidienne. On utilise les restes successifs de la division entière par 2.



# Décimal -> binaire

- 1ère méthode : la division Euclidienne. On utilise les restes successifs de la division entière par 2.




$173_{10}$ => $10101101_2$

# Décimal -> binaire

- 2ère méthode : Décomposition en puissance de deux (ou division par puissance de 2)

On recherche successivement les plus grandes puissances de 2 dans le nombre:

- Exemple : $(173)_{10} = 128 + 45$

$= 128 + 32 + 13$

$= 128 + 32 + 8 + 5$

$= 128 + 32 + 8 + 4 + 1$

$\rightarrow 10101101$

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| | | | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

# Opération en binaire

- Les règles de posage des opérations classiques restent les mêmes en binaire : opérations chiffre par chiffre avec les règles de retenue.

- Exemple : l'addition

```
   11    11
  1110  0011
+
  0110  0110
-------------
1 0100  1001
```

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

# Opération en binaire

- Les règles de posage des opérations classiques restent les mêmes en binaire : opérations chiffre par chiffre avec les règles de retenue.

- Exemple : l'addition

$$\begin{array}{rrrr} & {\scriptstyle 1\,1} & {\scriptstyle 1\,1} & \\ & 1110 & 0011 & \rightarrow 227 \\ + & & & \\ & 0110 & 0110 & \rightarrow 102 \\ \hline & 1\ 0100 & 1001 & \rightarrow 329 \end{array}$$

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

# Conclusion

- Toutes les données d'un ordinateur sont stockées et gérées sous forme binaire qui représente la présence ou l'absence de courant.
- En binaire, les chiffres d'un nombre sont multipliés par des puissances de 2 (puissances de 10 en base décimale)
- Nous avons vu comment représenter un entier en nombre binaire mais est-ce suffisant?
- Dans le prochain cours nous verront comment représenter d'autres nombres ainsi que d'autres types de données.