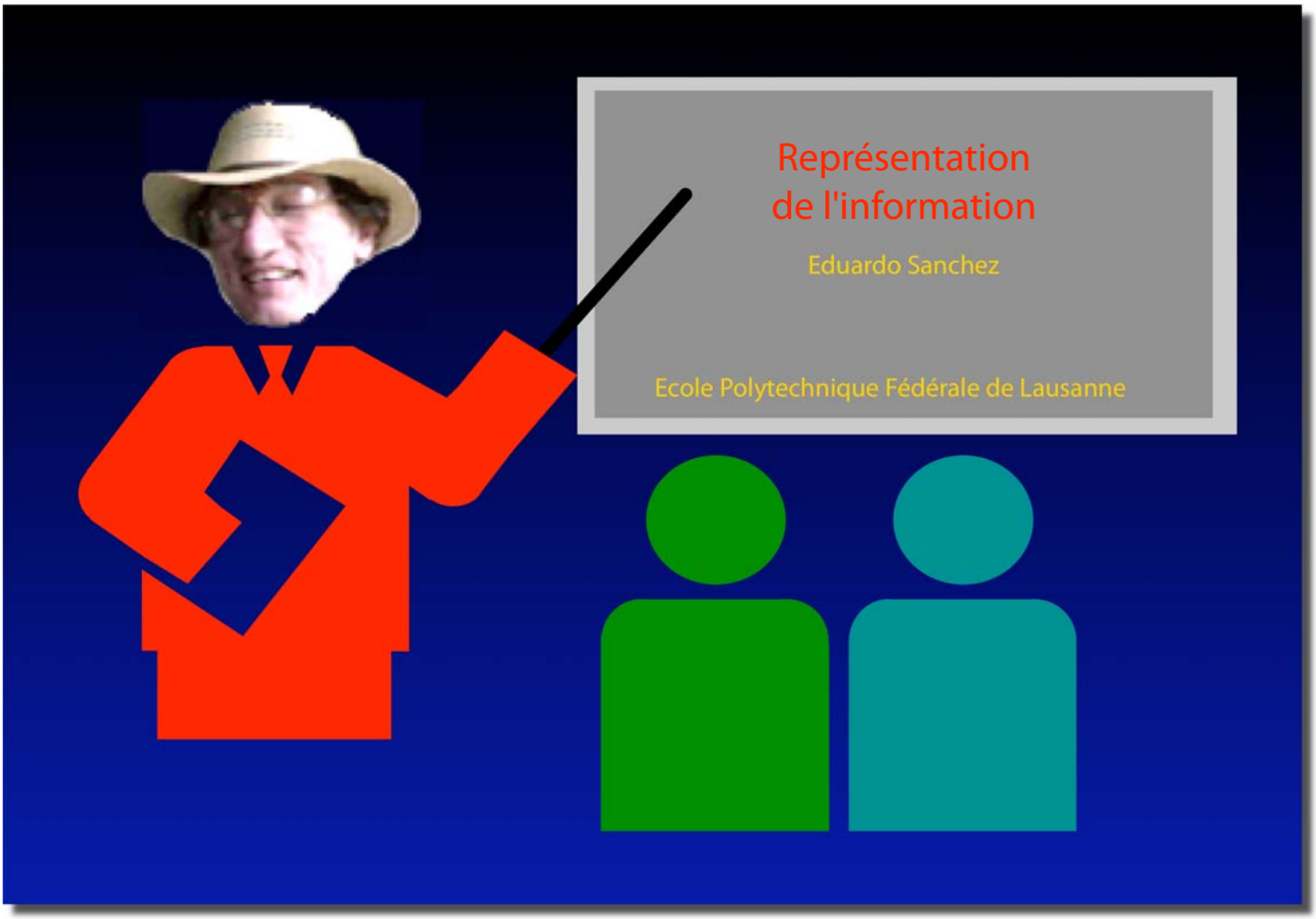
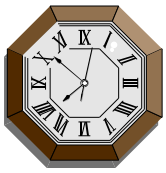



# Représentations de l'information

- Analogique:
  Les valeurs ne sont pas séparées par des sauts: entre deux valeurs A et B il existe un nombre infini d'autres valeurs

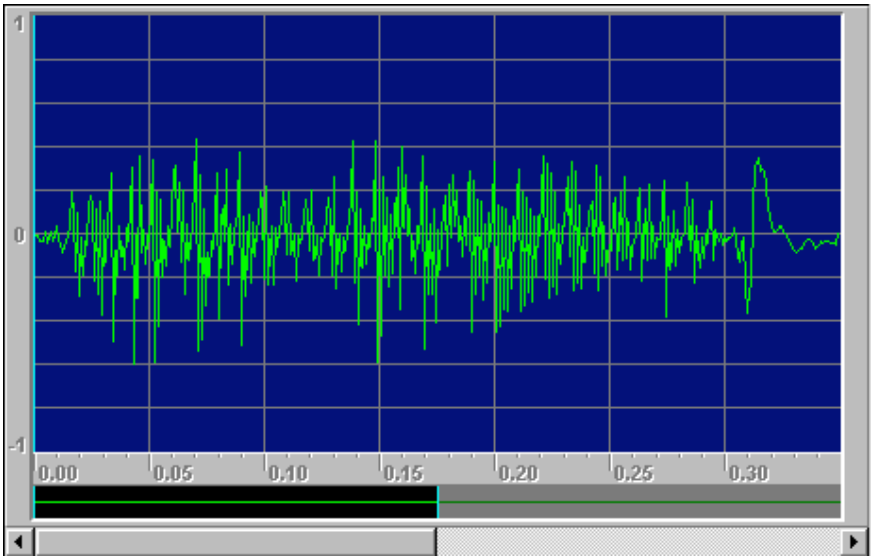


- Digitale (numérique):
  Une valeur est représentée par une chaîne finie de symboles appelés digits.
  Il est impossible de représenter digitalement tous les nombres existants entre deux points d'une échelle analogique

06:58

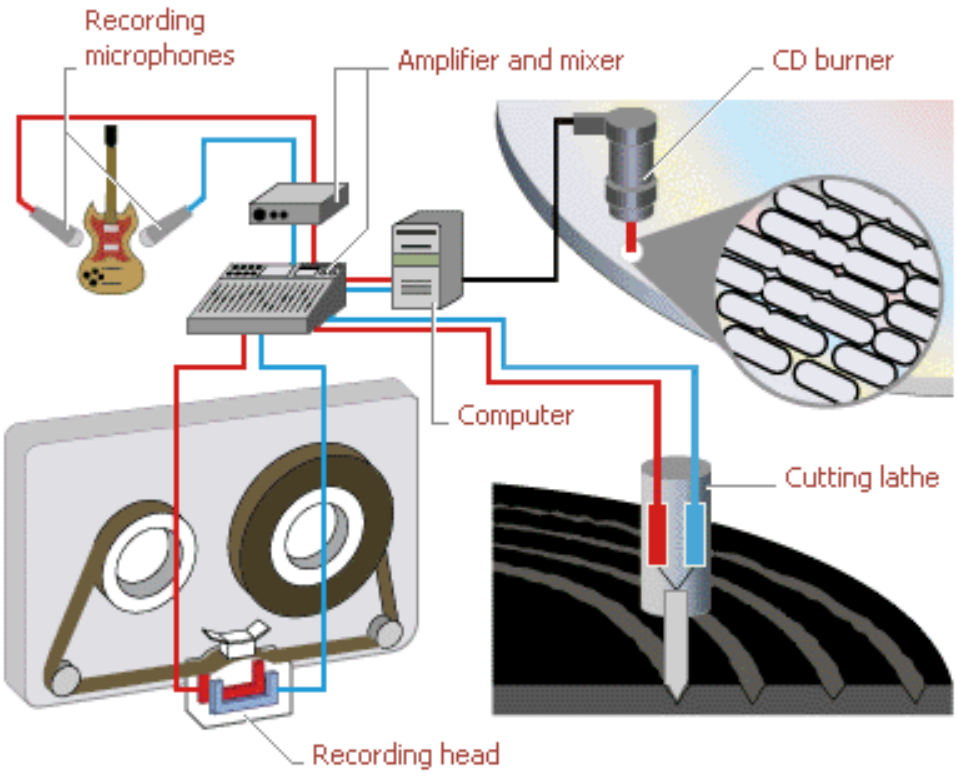

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

♦ Exemple: enregistrement analogique et digital du son



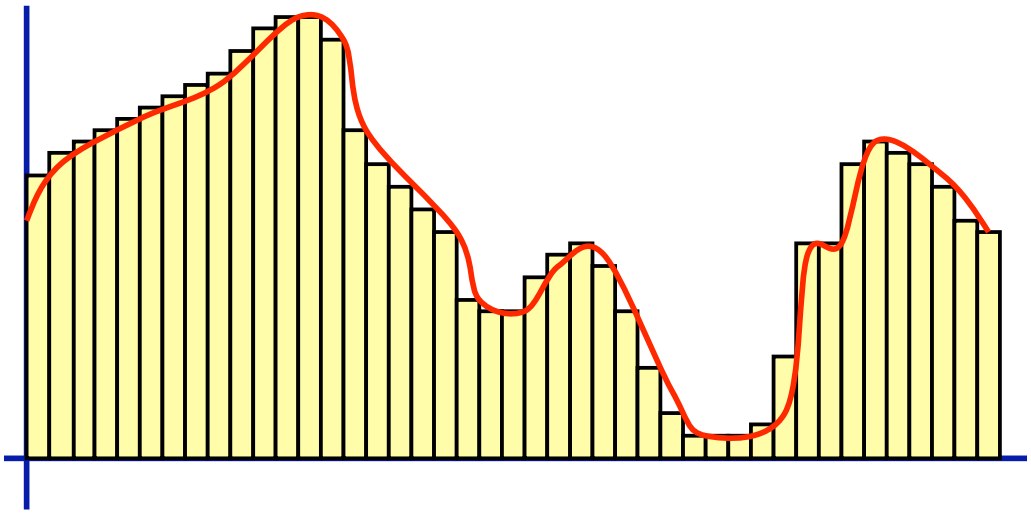



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne



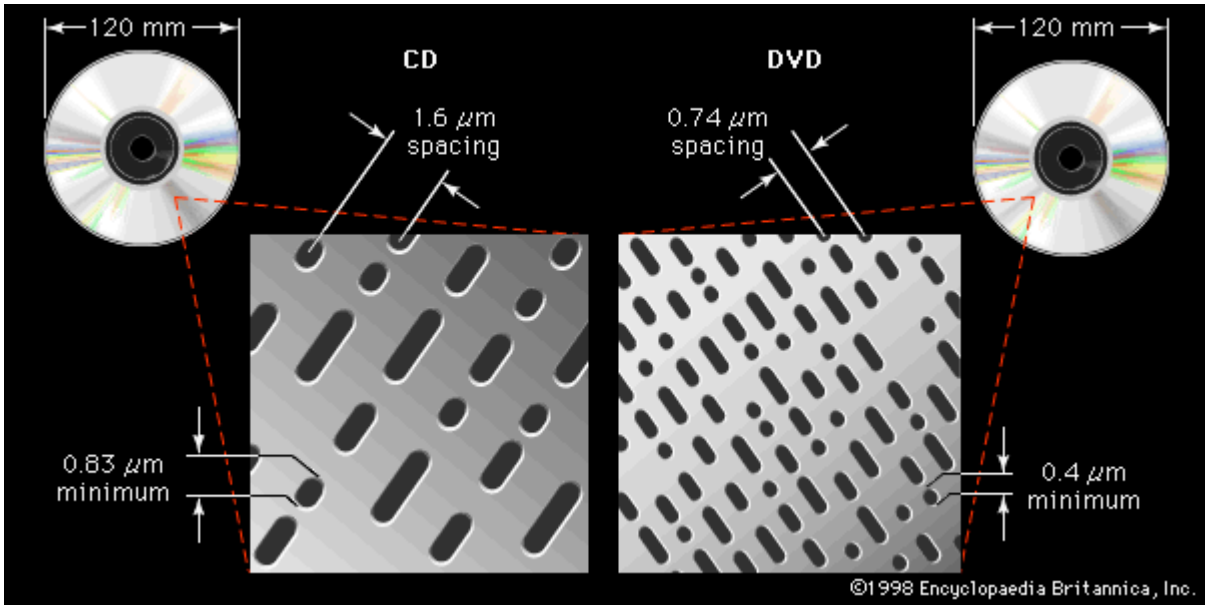



Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne



Pour enregistrer sur un CD, le son est échantillonné 44'100 fois par seconde. La valeur de chaque échantillon est stockée en binaire, à l'aide de 16 digits (*bits*): il n'y a que 65'536 valeurs possibles ($2^{16}$)


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne




Si le temps maximal d'enregistrement sur un CD est de 74 minutes, le nombre maximal de bits stockés dans un CD est donc de:

**(44100 échantillons/sec)*(16 bits)*(2 canaux)*(74*60 sec)=**
**6'265'728'000 bits = 783'216'000 *bytes***
**(1 byte = 8 bits)**


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Toute information dans un système informatique est représentée sous la forme d'un paquet de bits
- La différence entre un type d'information et un autre est donnée seulement par le contexte: la même séquence de bits peut représenter un nombre entier, un nombre réel, un caractère, une instruction, un son, etc

information = bits + contexte


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

# Représentation des nombres naturels

- Le système de numération romain a été heureusement remplacé par un système de numération de position dans une base choisie (normalement la base 10)
- Exemple:

$$\mathtt{MCMLIII} = 1953$$
$$1953 = 1\times10^3 + 9\times10^2 + 5\times10^1 + 3\times10^0$$
$$1953 = 1\times1000 + 9\times100 + 5\times10 + 3\times1$$
$$1953 = 1000 + 900 + 50 + 3$$


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Dans ce mode de représentation, en base n on utilise n symboles (chiffres) différents. Mais la valeur du chiffre change selon sa position
- Si un naturel X s'écrit en base $\beta$ sur N chiffres

  $x_{N-1}x_{N-2}\cdots x_1x_0$

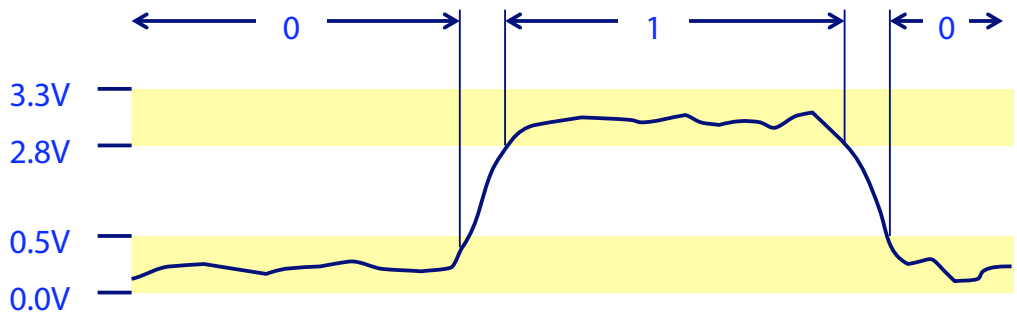
  la correspondance entre la valeur de X et celles des chiffres est donnée par l'équation:

$$X = \sum_{i=0}^{N-1} \beta^i x_i$$

- En informatique, on appelle $x_0$ le chiffre de poids faible (ou moins significatif), et $x_{N-1}$ le chiffre de poids fort (ou plus significatif)


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Si, dans la vie courante, nous utilisons la base 10, les ordinateurs utilisent la base 2
- Les problèmes de la base 10 sont:
  - difficulté de stockage
  - difficulté de transmission des 10 niveaux de signal nécessaires
  - difficulté d'implémentation des fonctions logiques et arithmétiques
- Par contre, la base 2 est facile à stocker, à l'aide d'éléments électroniques bistables, et sa transmission est fiable, même sur des environnements bruyants et imprécis





Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

# Exercices de conversion

- Passage de binaire à décimal:

$$01101011 = 0x2^7+1x2^6+1x2^5+0x2^4+1x2^3+0x2^2+1x2^1+1x2^0$$
$$= 64+32+8+2+1$$
$$= 107$$

- Passage d'hexadécimal à décimal:

$$A8CE = 10x16^3+8x16^2+12x16^1+14x16^0$$
$$= 40960+2048+192+14$$
$$= 43214$$


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Passage de décimal à binaire:

$193 = ?$

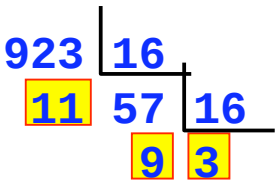
| | | |
|---|---|---|
| 193 | 2 | |
| 1 | 96 | 2 |
| 0 | 48 | 2 |
| 0 | 24 | 2 |
| 0 | 12 | 2 |
| 0 | 6 | 2 |
| 0 | 3 | 2 |
| 1 | 1 | |

$193 = 11000001_2$


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

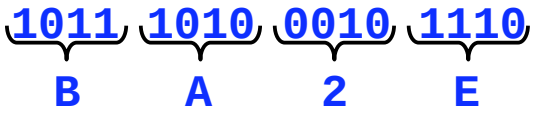
- Passage de décimal à hexadécimal:

**923 = ?**




**$923 = 39B_{16}$**

- Passage de binaire à hexadécimal:





Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

# Format de la représentation

- En base $\beta$, sur N chiffres, on peut représenter tous les naturels au sens large entre 0 et $\beta^N$-1
- Les ordinateurs ont un format pour représenter les nombres, c'est-à-dire un nombre de chiffres pré-établi. Pour cette raison, il est parfois utile d'écrire également les zéros à gauche
- Exemple: en base 2, sur 4 chiffres, on peut représenter les naturels entre 0 et 15

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


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Il est possible d'excéder la capacité de représentation d'un ordinateur (*overflow*), lors d'une addition par exemple

**10 + 9 = 19**

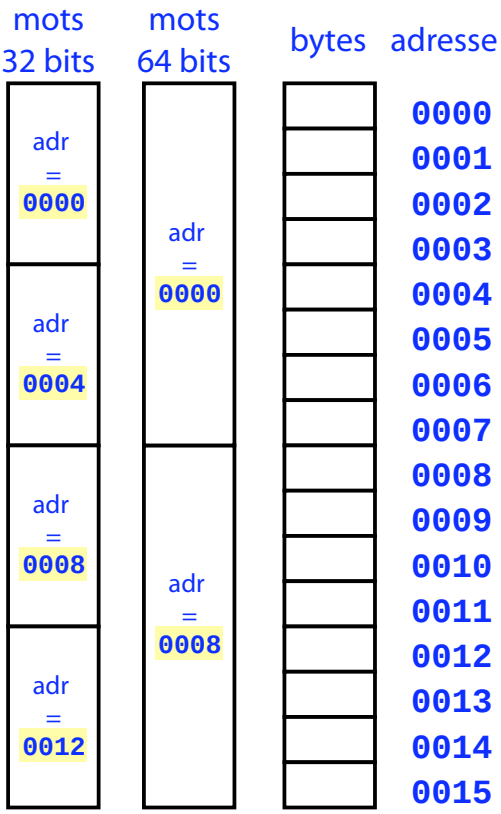
```
  1010
 +1001
 -----
 10011
```

bit perdu à cause du dépassement de capacité


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- On appelle généralement "taille d'un mot" le nombre de bits utilisés par un ordinateur pour stocker un entier
- La plupart des ordinateurs possèdent des mots 32 bits, bien que la tendance est d'aller vers les 64 bits
- Les ordinateurs peuvent travailler sur plusieurs formats de données, fractions ou multiples de la taille du mot. Toutefois, la taille de tous ces formats est toujours un multiple d'un byte
- Les données sont stockées dans une mémoire, chaque donnée à une adresse différente
- Chaque byte dans une mémoire possède une adresse différente. Si une donnée contient plus d'un byte, l'adresse de la donnée correspond à celle du premier byte


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne






Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Une adresse est stockée dans un mot de la mémoire d'un ordinateur
- Le nombre de bits d'un mot limite donc la taille maximale de la mémoire d'un ordinateur
- Si un ordinateur utilise des mots de 32 bits, la taille maximale de sa mémoire est de $2^{32}$ bytes, c'est-à-dire 4 gigabytes:

$$2^{32} = 2^2 x 2^{30} = 4GB$$



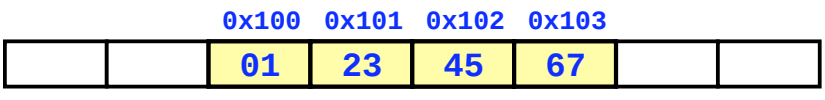
Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Les différents bytes d'un mot peuvent être ordonnés de différentes façons dans la mémoire
- Les deux ordonnancements les plus utilisés sont:
  - *big endian*:
    - le byte de poids fort est mis à l'adresse inférieure (le mot commence par le byte de poids fort)
    - utilisé par les ordinateurs Sun et Macintosh, par exemple
  - *little endian*:
    - le byte de poids faible est mis à l'adresse inférieure (le mot commence par le byte de poids faible)
    - utilisé par les ordinateurs PC et Alpha, par exemple

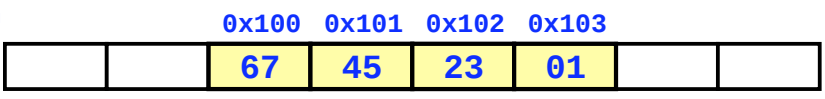

Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Exemple:
  - supposez que la valeur de la variable **`toto`** est **`0x01234567`** et
  - qu'elle est stockée à l'adresse **`0x0100`** (c'est-à-dire **`&toto=0x0100`**)

big endian




little endian



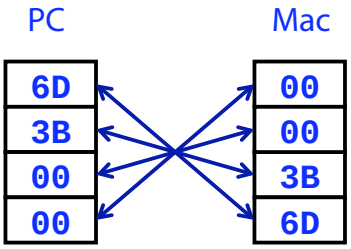


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Exemple, sur un ordinateur 32 bits:

```
int toto = 15213;
```

décimal: **15213**
binaire: **0011 1011 0110 1101**
hexadécimal: **3 B 6 D**





Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne

- Bien que, pour la plupart des cas, l'ordonnancement des bytes est transparent pour l'utilisateur, il existe des situations où il peut être à l'origine des erreurs:
  - lors de la transmission de données entre deux ordinateurs: le protocole de communication doit spécifier l'ordre de transmission des bytes
  - lors de l'examen d'un programme en assembleur (debugging, par exemple)
  - lors du traitement des données à bas niveau, possible avec des langages tels que C


Eduardo Sanchez
Ecole Polytechnique Fédérale de Lausanne