# Représenter des entiers en binaire

GIF-1001 Ordinateurs : Structure et Applications

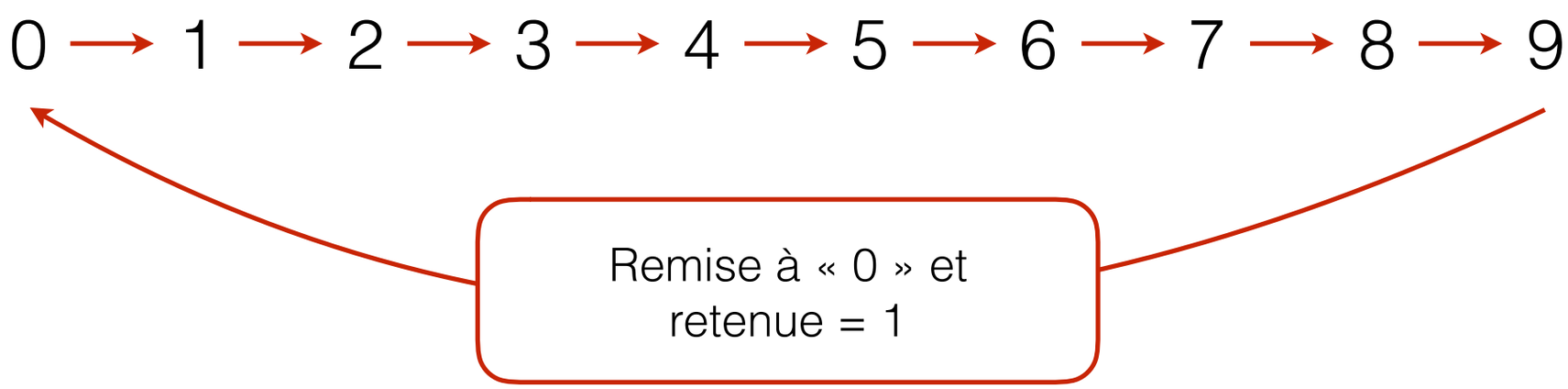
Jean-François Lalonde

# Représentation des entiers

- Chaque nombre doit avoir une représentation différente
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14…
  - I, II, III, IV, V, VI, VII, VIII, IX, X, XI, XII, XIII, XIV…
- Pour se simplifier la vie, on utilise une représentation avec laquelle il est facile de compter
  - CMXCIX + XVI = ?

# Compter en base 10 (décimal)

- 10 symboles: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Comptons: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, …?
- Que faire quand on n'a plus de symboles?
  - on recommence au début en ajoutant une retenue de 1 au prochain symbole



# Compter en base 10 (décimal)

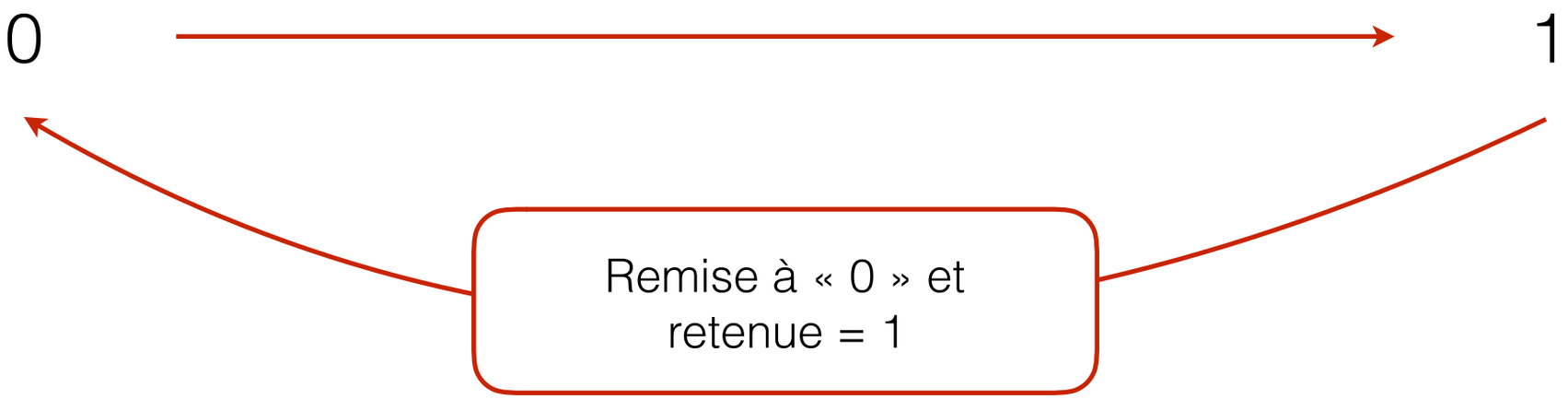
- Dans un nombre en base 10, chaque position correspond à une **puissance de 10**

| | | | | |
|---|---|---|---|---|
| Représentation décimale | Position 3 | Position 2 | Position 1 | Position 0 |
| | 1 | 4 | 2 | 3 |

Valeur décimale $1 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 = 1423$

# Compter en base 2 (binaire)

- 2 symboles: « 0 » et « 1 »
- Comptons: 0, 1, …?
- Que faire quand on n’a plus de symboles?
  - on recommence au début en ajoutant une retenue de 1 au prochain symbole



# Compter en base 2 (binaire)

- Dans un nombre en base 2, chaque position correspond à une **puissance de 2**

| Représentation décimale | Position 3 | Position 2 | Position 1 | Position 0 |
|---|---|---|---|---|
| | 1 | 0 | 1 | 1 |

Valeur décimale : $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

# Compter en base 2 (binaire)

- Dans un nombre en base 2, chaque position correspond à une **puissance de 2**

| Représentation décimale | Position 3 | Position 2 | Position 1 | Position 0 |
|---|---|---|---|---|
| | 1 | 0 | 1 | 1 |

Valeur décimale : $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

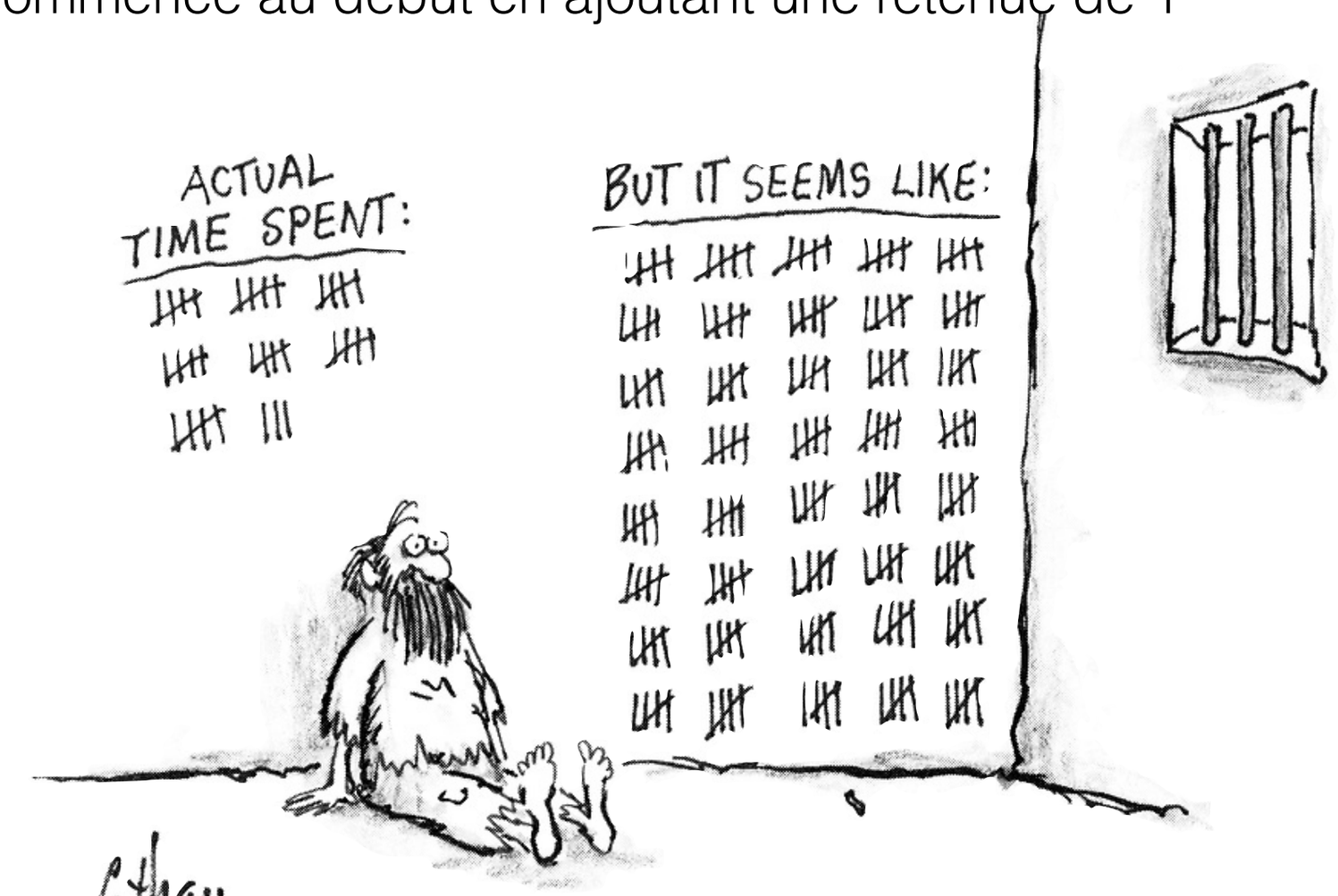
bit le **plus** significatif (il vaut $2^3$)

bit le **moins** significatif (il vaut $2^0$)

- Chaque symbole est nommé «bit».

# (détour): compter en base 1

- 1 symbole: 1
- Comptons: (rien), 1, …
- Que faire quand on n’a plus de symboles?
  - On recommence au début en ajoutant une retenue de 1



# (détour): compter en base 1

- Dans un nombre en base 1, chaque position correspond à une **puissance de 1 (donc 1)**

| | | | | |
|---|---|---|---|---|
| Représentation décimale | Position 3 | Position 2 | Position 1 | Position 0 |
| | 1 | 1 | 1 | 1 |

Valeur décimale $1 \times 1^3 + 1 \times 1^2 + 1 \times 1^1 + 1 \times 1^0 = 4$

# Récapitulation

- Pour représenter un nombre entier, nous sommes familiers avec la notation décimale, mais plusieurs options sont possibles.

- Il faut définir:

| Base | Symboles | |
|---|---|---|
| 1 | 1 | |
| 2 | 0 et 1 | (binaire) |
| 10 | 0 à 9 | (décimal) |

# Combien de nombres différents peut-on générer?

- Décimal
  - 2 symboles?
  - 3 symboles?
- Binaire
  - 4 symboles (bits)?
  - 8 symboles (bits)?
  - 16 symboles (bits)?

# Combien de nombres différents peut-on générer?

- Décimal
  - 2 symboles? ➛ 00 à 99 = $10^2$ = 100
  - 3 symboles? ➛ 000 à 999 = $10^3$ = 1000
- Binaire
  - 4 symboles (bits)? ➛ 0000 à 1111 = $2^4$ = 16
  - 8 symboles (bits)? ➛ 00000000 à 11111111 = $2^8$ = 256
  - 16 symboles (bits)? ➛ 0000… à 1111… = $2^{16}$ = 65536

# 4 PHIR™

(**P**oint **H**yper **I**mportants à **R**etenir)

Image source

# PHIR™ #1

- Dans un ordinateur, **tout, absolument tout**, est stocké en format binaire
  - entiers naturels (strictement positifs): $\mathbb{N}$
  - entiers relatifs (positifs et négatifs): $\mathbb{Z}$
  - réels (avec précision limitée): $\mathbb{R}$
  - chaîne de caractères: Bonjour!
  - programmes: `while (true) {printf(“Hello world!”);}`
  - images:

  - vidéos
  - twitter
  - facebook
- Bref, **tout**!

image

# PHIR™ #2

- Dans un ordinateur, on utilise un nombre **fini** de bits pour représenter de l'information.

- Ce nombre doit toujours être **pré-déterminé**, donné à l'avance par quelqu'un (le prof, le programmeur, un standard, etc…)

  - Exemple (par la programmeuse)

```
int toto = 2;
```

    - Le « `int` » indique (la plupart du temps) un entier relatif sur 32 bits

- Exemple (par le prof)
  - Convertissez 9 en binaire **sur 8 bits** ➔ 00001001

- Exemple (par le standard IEEE754)
  - L'exposant est stocké avec 8 bits.

Qu'est-ce que l'IEEE754?
Réponse au prochain cours…

# Nombre de bits

- En *théorie*, on pourrait utiliser le nombre de bits que l'on veut: 7, 13, 42, etc.

- En *pratique*, certains nombres sont plus communs:

  - 1 bit : 2 valeurs (0 et 1)
  - 4 bits: 16 valeurs (0 à 15)
  - 8 bits: 256 valeurs (0 à 255)
    - aussi appelé *octet*, ou, en anglais, *byte*
  - 16 bits: 65 536 valeurs (0 à 65 535)

# Bits vs octets

8 bits = 1 octet (*byte*)

- Combien de valeurs peut-on représenter avec 1 octet?
  - 1 octet = 8 bits = $2^8$ valeurs = 256
- Multiples d'octets:

« o » est l'abbréviation d'octet

  - 1 Kilo-octet (1 Ko) = $2^{10}$ o = 1024 o = 8 192 bits
  - 1 Mega-octet (1 Mo) = $2^{10}$ Ko = $2^{20}$ o = 8 388 608 bits
  - 1 Giga-octet (1 Go) = $2^{10}$ Mo = $2^{20}$ Ko = $2^{30}$ o = 8 589 934 592 bits
  - 1 Tera-octet (1 To) = $2^{10}$ Go = $2^{20}$ Mo = $2^{30}$ Ko = $2^{40}$ o = …
  - 1 Peta-octet (1 Po) = $2^{10}$ To = $2^{20}$ Go = $2^{30}$ Mo = $2^{40}$ Ko = $2^{50}$ o = …

# Conventions d’écriture

- Comment différencier
  - 1111 (binaire),
  - et 1111 (décimal)?

dans le cours nous privilégierons le préfixe 0b

- Binaire: on utilise le préfixe « 0b » ou l’indice « b ». Ex:
  - $0b1111 = 1111_b = 15$
- Décimal: aucune notation particulière.

# Représentation des entiers

**Entiers non-signés**

positifs (ou nuls)

0, 1, 2, …

entiers naturels ($\mathbb{N}$)

**Entiers signés**

positifs ou négatifs

… -2, -1, 0, 1, 2, …

entiers relatifs ($\mathbb{Z}$)

# Représentation des entiers

**Entiers non-signés**

positifs (ou nuls)

0, 1, 2, …

entiers naturels ($\mathbb{N}$)

**Entiers signés**

positifs ou négatifs

… -2, -1, 0, 1, 2, …

entiers relatifs ($\mathbb{Z}$)

# Conversion vers décimal

- binaire ➙ décimal
  - 0b10010101

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 0 | 0 | 16 | 0 | 4 | 0 | 1 |

= 149

# Conversion vers décimal

- binaire ➙ décimal
  - 0b10010101

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 0 | 0 | 16 | 0 | 4 | 0 | 1 |

= 149

  - 0b11001011

# Conversion vers décimal

- binaire ➙ décimal
  - 0b10010101

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 0 | 0 | 16 | 0 | 4 | 0 | 1 |

= 149

  - 0b11001011

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | | | | | | | | |

# Conversion vers décimal

- binaire ➙ décimal
  - 0b10010101

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 0 | 0 | 16 | 0 | 4 | 0 | 1 |

= 149

  - 0b11001011

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 64 | 0 | 0 | 8 | 0 | 2 | 1 |

= 203

# Conversion: décimal ➙ binaire (approche 1)

- 10 = 0b?

| | | | |
|---|---|---|---|
| 10 | 2 | | |
| -10 | 5 | 2 | |
| **0** | -4 | 2 | 2 |
| | **1** | -2 | **1** |
| | | **0** | |

- 10 = 0b1010

Selon moi, cette approche est plus simple, mais utilisez votre préférée!

# Conversion: décimal ➙ binaire (approche 2)

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

- 10 en binaire sur 8 bits?

Selon moi, cette approche est plus simple, mais utilisez votre préférée!

# Conversion: décimal ➙ binaire (approche 2)

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

- 10 en binaire sur 8 bits?
  - commencer de la gauche: 0 x 128, 0 x 64, …

| Bits | 0 | 0 | 0 | 0 | | | | |
|---|---|---|---|---|---|---|---|---|

- 1 x 8. On met un “1” à la position 3. Reste 2

| Bits | 0 | 0 | 0 | 0 | 1 | | | |
|---|---|---|---|---|---|---|---|---|

  - 0 x 4

| Bits | 0 | 0 | 0 | 0 | 1 | 0 | | |
|---|---|---|---|---|---|---|---|---|

  - 1 x 2. On met un “1” à la position 1. Reste 0. Terminé!

| Bits | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|

# Conversion: décimal ➙ binaire (approche 2)

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

- 147 en binaire sur 8 bits?

# Conversion: décimal ➙ binaire (approche 2)

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

- 147 en binaire sur 8 bits?
  - commencer de la gauche: 1 x 128. Reste 19

| Bits | 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|

  - 0 x 64, 0 x 32, 1 x 16. Reste 3

| Bits | 1 | 0 | 0 | 1 | | | | |
|---|---|---|---|---|---|---|---|---|

  - 0 x 8, 0 x 4, 1 x 2. Reste 1

| Bits | 1 | 0 | 0 | 1 | 0 | 0 | 1 | |
|---|---|---|---|---|---|---|---|---|

  - 1 x 1. Terminé!

| Bits | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|

# Addition en binaire

Cas simples

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

dépassements

Cas à plusieurs bits

retenues

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

# Représentation des entiers

**Entiers non-signés**

positifs (ou nuls)

0, 1, 2, …

entiers naturels ($\mathbb{N}$)

**Entiers signés**

positifs ou négatifs

… -2, -1, 0, 1, 2, …

entiers relatifs ($\mathbb{Z}$)

# Représentation des entiers

**Entiers non-signés**

positifs (ou nuls)

0, 1, 2, …

entiers naturels ($\mathbb{N}$)

**Entiers signés**

positifs ou négatifs

… -2, -1, 0, 1, 2, …

entiers relatifs ($\mathbb{Z}$)

# Nombres entiers signés

- Premier essai: représentation « signe et magnitude »
  - Le premier bit (à gauche) est le **signe**: 0 = positif, 1 = négatif
  - Le reste est la **magnitude**
  - Exemple (4 bits): $1101_b = (-1) \times (4 + 1) = -5$
  - Combien de nombres peut-on représenter?

# Nombres entiers signés

- Problèmes?
  - Représentation de 0?
    - $0 = 0000_b = 1000_b$
  - Opérations arithmétiques:
    - $-3 + 4 = 1011_b + 0100_b = 1111_b = -7$!

# Représentation « complément 2 »

Entiers signés

- Seule différence: le premier bit représente $-2^{N-1}$

« N » est le nombre de bits

- Exemple: $1101_b$ = ?
  - $1101_b = -2^3 + 2^2 + 2^0 = -3$.

# Représentation « complément 2 »

Entiers signés

- Seule différence: le premier bit représente $-2^{N-1}$

« N » est le nombre de bits

- Exemple: $1101_b$ = ?
  - $1101_b = -2^3 + 2^2 + 2^0 = -3$.
- Pour changer le signe d'un nombre, il faut:
  - le soustraire de $2^N$ (d'où le nom « complément 2 »)
  - plus direct (et plus facile): inverser tous les bits et ajouter 1.

# Représentation « complément 2 »

Entiers signés

- Seule différence: le premier bit représente $-2^{N-1}$

« N » est le nombre de bits

- Exemple: $1101_b$ = ?
  - $1101_b = -2^3 + 2^2 + 2^0 = -3$.
- Pour changer le signe d'un nombre, il faut:
  - le soustraire de $2^N$ (d'où le nom « complément 2 »)
  - plus direct (et plus facile): inverser tous les bits et ajouter 1.
- Exemple: $3 = 0011_b$. $-3 = ?_b$
  - Si on prend le complément + 1, on obtient: $1100_b + 1_b = 1101_b$.

# Représentation « complément 2 »



- Combien de nombres peut-on représenter?

# Représentation « complément 2 »

- Problèmes de tout à l'heure?
  - Représentation de 0?
    - $0000_b$ seulement (maintenant, $1000_b = -8_d$)
  - Opérations arithmétiques:
    - $-3 + 4 = 1101_b + 0100_b = 0001_b = 1$ (ouf!)

# Représentation « complément 2 »

- Et la soustraction?
  - En pratique, on peut soustraire avec une addition avec l'inverse (en complément 2).
  - Par ex.: 2 - 3 = 2 + (-3) = -1.

Nous utiliserons **toujours** le « complément 2 » pour représenter les entiers signés.

Cela est valable pour le cours, mais en pratique également!

# Non-signé vs signé

- Non-signé
  - 0b11001011

Entiers non-signés

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Valeur | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | 128 | 64 | 0 | 0 | 8 | 0 | 2 | 1 |

= 203

- Signé (complément 2)
  - 0b11001011

Entiers signés

| Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Valeur | -128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| = | -128 | 64 | 0 | 0 | 8 | 0 | 2 | 1 |

= -53

# Non-signé vs signé

Entiers non-signés
(positifs ou nuls)

Entiers signés
(positifs ou négatifs)

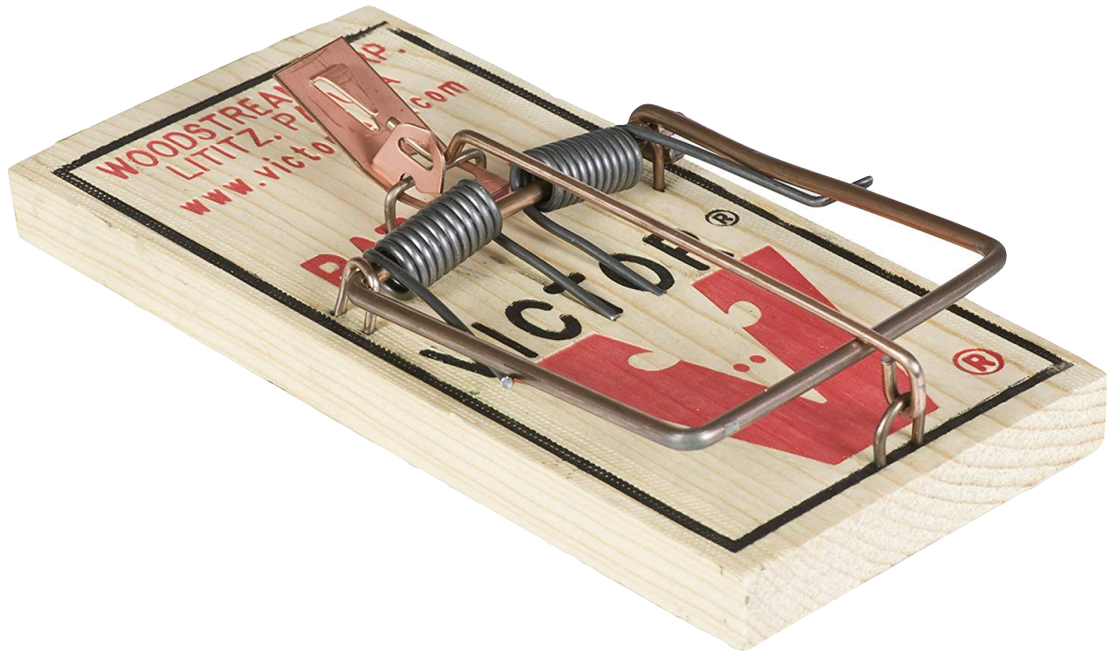
*Seule différence:*

Bit le plus significatif
(le plus à gauche)

$2^{N-1}$

$-2^{N-1}$

# Question (piège)




Quelle est la valeur décimale de 0b1010?

# PHIR™ #3

- A priori, **nous ne pouvons pas savoir** ce qu'une chaîne binaire signifie.
  - Ex: quelle est la valeur décimale de 0b1010?
  - La bonne réponse est: ça dépend!

positifs (ou nuls)

| | |
|---|---|
| entier non-signé | 10 |
| entier signé | -6 |

positifs ou négatifs

- Il faut donc savoir **quel format utiliser** pour bien interpréter les données

# Cas spécial 1: débordement (« overflow »)

- Quels nombres peut-on représenter sur 4 bits en complément 2?
  - -8 à +7
- Qu'arrive-t-il si on additionne (sur 4 bits)
  - 5 + 4 = ?
    - on obtient -7!
- Comment faire pour détecter un débordement?
  - le bit de signe change

# Cas spécial 2: retenue (« carry »)

- Condition: s'il y a une retenue qui « dépasse » des bits les plus significatifs
- Utile pour l'arithmétique *non-signée* (pas important pour l'arithmétique signée)
- Ex: additionner 10+7 sur 4 bits