Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Hes·so

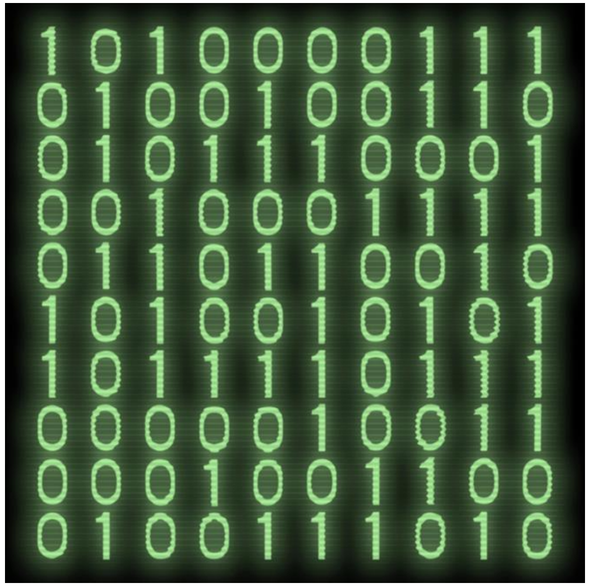
Téléinformatique – Ch. 3

Binaire, hexadécimal et opérateurs logiques

Vincent Magnin
vincent.magnin@hefr.ch

Objectifs

- Connaître les unités couramment utilisées dans l'informatique.
- Être capable de représenter et de convertir des nombres entre différentes bases.
- Pouvoir effectuer des opérations logiques sur des nombres de différentes bases.



Termes couramment utilisés

Le bit :

- Chiffre binaire. Le plus petit élément d'information en informatique. Les valeurs possibles sont 0 et 1.
- Notation : b

Le byte :

- Groupement de chiffres binaires. La taille standard est 8 bits, mais ce n'est pas obligatoire. On utilise souvent le terme octet pour éviter toute ambiguïté.
- Notation : B

L'octet :

- Groupement de 8 bits.
- Notation : o

Tailles couramment utilisées

• Kilo	10^3	1'000
• Méga	10^6	1'000'000
• Giga	10^9	1'000'000'000
• Téra	10^{12}	1'000'000'000'000

On parle alors de :

- Kilobit (kb)
- Mégabit (Mb)
- MégaByte (MB)
- MégaOctet (Mo)
- TéraByte (TB)

- Ne pas confondre avec le kibibit : 1 Kibit = 2^{10} = 1024

Représentation des nombres

Historiquement (et naturellement), l'humain a toujours compté en base 10 (principalement pour les 10 doigts que l'on possède). Cette base s'appelle la base décimale.

- 10 symboles : {0,1,2,3,4,5,6,7,8,9}
- La position du chiffre (symbole) influence son poids
- Exemple :
  - $768 = 7*10^2 + 6*10^1 + 8*10^0$
  - $768 = 7*100 + 6*10 + 8*1$

Chiffre (symbole, multiplicateur)
Base
Puissances successives

Base et puissances successives : Poids

La base décimale est arbitraire. Il est possible de représenter les nombres avec d'autres bases.

Représentation des nombres (2)

La base 2 (binaire) est celle utilisée par les ordinateurs. On dispose de 2 symboles : {0,1} pour écrire les nombres. Comme pour la base décimale, la position de chaque symbole influence son poids. On indique les nombres binaires avec le préfixe 0b. Chaque symbole représente un bit.

Exemple avec le nombre 12 :

$$12_d = 0b1100$$

$$12_d = 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0$$

$$12_d = 8 + 4$$

- Chiffre (symbole, multiplicateur)
- Base
- Puissances successives

$\left.\begin{array}{l}\text{Base}\\ \text{Puissances successives}\end{array}\right\}$ Poids

Exemple avec le nombre 53 :

$$53_d = 0b110101$$

$$53_d = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$$

$$53_d = 32 + 16 + 4 + 1$$

Représentation des nombres (3)

La base 16 (hexadécimale) est un moyen de représenter les nombres avec 16 symboles : {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}. Ici, A = 10, B = 11, C = 12, D = 13, E = 14 et F = 15. On indique les nombres hexadécimaux avec les préfixes 0h ou 0x.

Cette base est très utilisée car la conversion binaire ↔ hexadécimal est assez facile. Elle représente très bien les informations. 1 octet (8 bits) = une valeur hexadécimale à 2 symboles.

Exemple avec 31 :

31_d = 0b11111

31_d = 0x1F

Exemple avec 763 :

763_d = 0b1011111011

763_d = 0x2FB

Table de correspondance

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Opérateurs logiques

Les opérateurs logiques sont utilisés pour manipuler les nombres binaires. Il existe de nombreux opérateurs logiques. Voici les principaux :

Opérateur **AND** (ET) : intersection logique (le produit).

S = A • B ou A & B ou A AND B

Opérateur **OR** (OU) : réunion logique (la somme).

S = A+B ou A | B ou A OR B

Opérateur **XOR** (OU exclusif) : seule une entrée doit être vraie.

S = A ⊕ B ou A ^ B ou A XOR B

Opérateur **NOT** : Inversion.

S = $\bar{A}$ ou !A ou NOT A

A	$\bar{A}$
0	1
1	0

Table de vérité

Une table de vérité décrit, pour une équation logique donnée, tous les résultats possibles en fonction des valeurs d'entrée.

A	B	A • B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Opérations logiques

Soit 2 nombres A et B, codés en binaire :

A = 0b01101101

B = 0b10011100

On peut manipuler ces 2 nombres avec les opérateurs étudiés précédemment :

	AND	OR	XOR
A	01101101	01101101	01101101
B	10011100	10011100	10011100
Résultat	00001100	11111101	11110001

Conversion binaire → hexadécimal

<table>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td colspan="4">9</td><td colspan="4">C</td><td colspan="4">6</td><td colspan="4">D</td></tr>
</table>

…

$0110\ 1101_b$

0110_b
$= 6_d$
$= 6_h$

1101_b
$= 13_d$
$= D_h$

$\rightarrow = 6D_h$

Conversion hexadécimal → binaire

<table>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td colspan="4">9</td><td colspan="4">C</td><td colspan="4">6</td><td colspan="4">D</td></tr>
</table>

$9C_h$ ...

9_h
$= 9_d$
$= 1001_b$

C_h
$= 12_d$
$= 1100_b$

→ $= 10011100_b$

Références

- Ancien cours « Téléinformatique » (G. Waeber, S. Paccard, Q. Vaucher, N. Wirth).
- Ancien cours « Téléinformatique » (M. Roch-Neirey).