# *Numération et codage (rappels)*

Vincent Risch, septembre 2008, révision septembre 2013

I.U.T., Aix-Marseille Université

# *Introduction*

- L'ordinateur traite de l'information digitale ; le support de l'information est système à deux états d'équilibre $0$ et $1$ appelés bits.

# *Introduction*

- L'ordinateur traite de l'information digitale ; le support de l'information est système à deux états d'équilibre $0$ et $1$ appelés bits.
- A $n$ bits correspondent $2^n$ configurations binaires possibles différentes.

# *Introduction*

- L'ordinateur traite de l'information digitale ; le support de l'information est système à deux états d'équilibre $0$ et $1$ appelés bits.
- A $n$ bits correspondent $2^n$ configurations binaires possibles différentes.
- Mot : séquence $q_{n-1} \ldots q_0$ de $n$ bits ; $q_{n-1}$ est appelé bit de poids fort, $q_0$ bit de poids faible.

# *Introduction*

- L'ordinateur traite de l'information digitale ; le support de l'information est système à deux états d'équilibre $0$ et $1$ appelés bits.
- A $n$ bits correspondent $2^n$ configurations binaires possibles différentes.
- Mot : séquence $q_{n-1} \ldots q_0$ de $n$ bits ; $q_{n-1}$ est appelé bit de poids fort, $q_0$ bit de poids faible.
- Octet : mot de huit bits.

# *Introduction*

- L'ordinateur traite de l'information digitale ; le support de l'information est système à deux états d'équilibre $0$ et $1$ appelés bits.
- A $n$ bits correspondent $2^n$ configurations binaires possibles différentes.
- Mot : séquence $q_{n-1} \ldots q_0$ de $n$ bits ; $q_{n-1}$ est appelé bit de poids fort, $q_0$ bit de poids faible.
- Octet : mot de huit bits.
- Code : correspondance biunivoque entre les informations à représenter et les configurations binaires possibles.

# *Le codage des caractères (1)*

- Code ASCII : données alphanumériques sur sept bits plus un bit de parité.

# *Le codage des caractères (1)*

- Code ASCII : données alphanumériques sur sept bits plus un bit de parité.
- Standard ISO 8859-1 (Latin 1) : évolution du code ASCII, huit bits pour représenter entre autres les caractères accentués.

# *Le codage des caractères (1)*

- Code ASCII : données alphanumériques sur sept bits plus un bit de parité.
- Standard ISO 8859-1 (Latin 1) : évolution du code ASCII, huit bits pour représenter entre autres les caractères accentués.
- Unicode : deux octets pour coder des jeu de caractères non latins.

# *Le codage des caractères (2)*

## Table ASCII

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 0000 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0010 | STX | DC2 | " | 2 | B | R | b | r |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0101 | ENQ | NAK | % | 5 | E | U | e | u |
| 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 0111 | BEL | ETB | ’ | 7 | G | W | g | w |
| 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 1001 | HT | EM | ) | 9 | I | Y | i | y |
| 1010 | LF | SUB | * | : | J | Z | j | z |
| 1011 | VT | ESC | + | ; | K | [ | k | { |
| 1100 | FF | FS | , | < | L | \ | l | \| |
| 1101 | CR | GS | - | = | M | ] | m | } |
| 1110 | SO | RS | . | > | N | ^ | n | ~ |
| 1111 | SI | US | / | ? | O | _ | o | DEL |

# *Codage des entiers naturels : préliminaires*

**Lemme 1**

$$(\forall a \in \mathbb{N})(\forall b \in \mathbb{N}^*)(\exists k \in \mathbb{N}^*)(a < k.b)$$

# *Codage des entiers naturels : préliminaires*

**Lemme 1**

$$(\forall a \in \mathbb{N})(\forall b \in \mathbb{N}^*)(\exists k \in \mathbb{N}^*)(a < k.b)$$

**Théorème 1** *Quels que soient* $a \in \mathbb{N}$ *et* $b \in \mathbb{N}^*$*, il existe un couple unique* $(q, r) \in \mathbb{N} \times \mathbb{N}$ *tels que* $a = b.q + r$ *et* $0 \leq r < b$*.*

# *Codage des entiers naturels : préliminaires*

**Lemme 1**

$$(\forall a \in \mathbb{N})(\forall b \in \mathbb{N}^*)(\exists k \in \mathbb{N}^*)(a < k.b)$$

**Théorème 1** *Quels que soient* $a \in \mathbb{N}$ *et* $b \in \mathbb{N}^*$*, il existe un couple unique* $(q, r) \in \mathbb{N} \times \mathbb{N}$ *tels que* $a = b.q + r$ *et* $0 \leq r < b$*.*

On a effectué la division euclidienne de $a$ par $b$, $q$ est le quotient, $r$ le reste.

# *Décomposition polynômiale*

**Théorème 2** *Soit $b \in \mathbb{N}, b > 1$. Pour tout $a \in \mathbb{N}$, supérieur ou égal à $b$, il existe une séquence unique de $n$ entiers naturels $q_0 \ldots q_{n-1}$ tels que :*

$$a = q_{n-1}.b^{n-1} + \ldots + q_2.b^2 + q_1.b + q_0$$

*avec* $0 < q_{n-1} < b$ ***et*** $\forall i \in \{0, 1, \ldots, (n-2)\}, 0 \leq q_i < b$.

# *Décomposition polynômiale*

**Théorème 2** *Soit* $b \in \mathbb{N}, b > 1$. *Pour tout* $a \in \mathbb{N}$, *supérieur ou égal à* $b$, *il existe une séquence unique de* $n$ *entiers naturels* $q_0 \ldots q_{n-1}$ *tels que :*

$$a = q_{n-1}.b^{n-1} + \ldots + q_2.b^2 + q_1.b + q_0$$

*avec* $0 < q_{n-1} < b$ ***et*** $\forall i \in \{0, 1, \ldots, (n-2)\}, 0 \leq q_i < b$.

La représentation du nombre $a$ dans la base $b$ s'écrit alors $q_{n-1}q_{n-2} \ldots q_{0_b}$ où la représentation de $b$ est elle-même donnée dans le système décimal. Par exemple, on note la représentation de l'entier $377$ par $377_{10}$ en base $10$, ou $4302_5$ en base $5$.

# *Décomposition binaire*

En ne disposant que de deux symboles ($0$ et $1$), la base $2$ s'impose comme le seul système de numération adapté à la représentation interne des entiers naturels.

**Corollaire 1 (Théorème 2)** *Tout $a \in \mathbb{N}$, supérieur ou égal à $2$, s'écrit de manière unique sous la forme :*

$$\begin{aligned} a &= q_{n-1}.2^{n-1} + \ldots + q_2.2^2 + q_1.2 + q_0 \\ &= \sum_{i=0}^{n-1} q_i.2^i \end{aligned}$$

*avec $q_{n-1} = 1$ et $q_i \in \{0, 1\}$ pour $0 \leq i \leq n-2$.*

# *Exemple*

L'entier $43_{10}$ se décompose en base $2$ :

$$43_{10} = 1.2^5 + 0.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0$$

Autrement dit,$43_{10}$ sécrit en binaire $101011_2$.

# *Résultats complémentaires (1)*

**Théorème 3** *Sur $k$ bits, on peut représenter les entiers naturels $a$ tels que $0 \leq a \leq 2^k - 1$.*

*Preuve :* Le plus grand entier représentable sur $k$ bits est :

$$\sum_{i=0}^{k-1} 2^i = 2^k - 1$$

*Q.E.D.*

- Sur un octet on compte de $0$ à $255$.
- Sur deux octets on compte de $0$ à $32\ 767$.
- Un mot de $32$ bits permet de compter de $0$ à $4\ 294\ 967\ 295$.

# *Résultats complémentaires (2)*

**Corollaire 2** *Pour tout entier naturel $a$, il existe un entier $k$ strictement supérieur à $0$, tel que $2^{k-1} \leq a < 2^k$.*

*Remarque :* La représentation binaire est malcommode à utiliser pour l'être humain : la longueur du codage et le manque de discrimination dû à l'utilisation de deux caractères seulement est facilement source d'erreurs. . .

# *Représentation hexadécimale*

La numération héxadécimale représente le meilleur compromis pour tout système informatique qui utlise des mots dont le nombre de bits est divisible par 4.

| Décimal | DCB | Héxadécimal |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | $A$ |
| 11 | 1011 | $B$ |
| 12 | 1100 | $C$ |
| 13 | 1101 | $D$ |
| 14 | 1110 | $E$ |
| 15 | 1111 | $F$ |

# *Décomposition hexadécimale*

**Corollaire 3 (Théorème 2)** *Tout $a \in \mathbb{N}$, supérieur ou égal à $2$, s'écrit de manière unique sous la forme :*

$$\begin{aligned} a &= q_{n-1}.16^{n-1} + \ldots + q_2.16^2 + q_1.16 + q_0 \\ &= \sum_{i=0}^{n-1} q_i.16^i \end{aligned}$$

*avec*

$$\begin{aligned} q_{n-1} &\in \{1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\} \quad \textit{et} \\ q_i &\in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\} \\ &\quad \textit{pour } 0 \leq i \leq n-2. \end{aligned}$$

# *Exemple*

L'entier $43_{10}$ se décompose en base $16$ :

$$43_{10} = 2.16^1 + B.16^0$$

Autrement dit, $43_{10}$ s'écrit en héxadécimal $2B_{16}$.

# *Conversion d'un entier naturel*

Deux méthodes :

1. la méthode des divisions successives ;
2. la méthode des soustractions successives.

La mise en œuvre de chacune de ces deux méthodes n'est qu'une variation à partir du résultat du théorème de décomposition polynômiale.

# *Méthode des divisions successives*

On a :

$$
\begin{aligned}
a &= q_{n-1}.b^{n-1} + q_{n-2}.b^{n-2} + \ldots + q_2.b^2 + q_1.b^1 + q_0 \\
&= \underbrace{\left(q_{n-1}.b^{n-2} + q_{n-2}.b^{n-3} + \ldots + q_2.b^1 + q_1\right)}_{a_1 \text{ (quotient)}}.b + \underbrace{q_0}_{\text{(reste)}}
\end{aligned}
$$

# *Méthode des divisions successives*

On a :

$$\begin{aligned} a &= q_{n-1}.b^{n-1} + q_{n-2}.b^{n-2} + \ldots + q_2.b^2 + q_1.b^1 + q_0 \\ &= \underbrace{(q_{n-1}.b^{n-2} + q_{n-2}.b^{n-3} + \ldots + q_2.b^1 + q_1)}_{a_1 \text{ (quotient)}}.b + \underbrace{q_0}_{\text{(reste)}} \end{aligned}$$

puis :

$$\begin{aligned} a_1 &= q_{n-1}.b^{n-2} + q_{n-2}.b^{n-3} + \ldots + q_2.b^1 + q_1 \\ &= \underbrace{(q_{n-1}.b^{n-3} + q_{n-2}.b^{n-4} + \ldots + q_2)}_{a_2 \text{ (quotient)}}.b + \underbrace{q_1}_{\text{(reste)}} \end{aligned}$$

# *Méthode des divisions successives*

On a :

$$\begin{aligned} a &= q_{n-1}.b^{n-1} + q_{n-2}.b^{n-2} + \ldots + q_2.b^2 + q_1.b^1 + q_0 \\ &= \underbrace{(q_{n-1}.b^{n-2} + q_{n-2}.b^{n-3} + \ldots + q_2.b^1 + q_1)}_{a_1 \text{ (quotient)}}.b + \underbrace{q_0}_{\text{(reste)}} \end{aligned}$$

puis :

$$\begin{aligned} a_1 &= q_{n-1}.b^{n-2} + q_{n-2}.b^{n-3} + \ldots + q_2.b^1 + q_1 \\ &= \underbrace{(q_{n-1}.b^{n-3} + q_{n-2}.b^{n-4} + \ldots + q_2)}_{a_2 \text{ (quotient)}}.b + \underbrace{q_1}_{\text{(reste)}} \end{aligned}$$

et ainsi de suite. . .

# *Algorithme*

- l'entier décimal $a$ est divisé par $b$, ce qui donne pour quotient $a_1$ et pour reste $q_0$ ;

# *Algorithme*

- l'entier décimal $a$ est divisé par $b$, ce qui donne pour quotient $a_1$ et pour reste $q_0$ ;
- si $a_1 > 0$, il est divisé par $b$, ce qui donne pour quotient $a_2$ et pour reste $q_1$ ;

# *Algorithme*

- l'entier décimal $a$ est divisé par $b$, ce qui donne pour quotient $a_1$ et pour reste $q_0$ ;
- si $a_1 > 0$, il est divisé par $b$, ce qui donne pour quotient $a_2$ et pour reste $q_1$ ;
- l'opération de division mentionnée ci-dessus est renouvelée sur chacun des quotients obtenus jusqu'à obtenir un quotient $a_m = 0$ ;

# *Algorithme*

- l'entier décimal $a$ est divisé par $b$, ce qui donne pour quotient $a_1$ et pour reste $q_0$ ;
- si $a_1 > 0$, il est divisé par $b$, ce qui donne pour quotient $a_2$ et pour reste $q_1$ ;
- l'opération de division mentionnée ci-dessus est renouvelée sur chacun des quotients obtenus jusqu'à obtenir un quotient $a_m = 0$ ;
- les restes obtenus successivement sont les chiffres de la représentation en base $b$ de l'entier $a$, avec $q_0$ comme chiffre le moins significatif, $q_1$ comme chiffre suivant, et ainsi de suite.

# *Exemple*

Conversion du nombre $43_{10}$ en binaire :

$$\begin{aligned} 43 &= 21.2 + 1 \\ 21 &= 10.2 + 1 \\ 10 &= 5.2 + 0 \\ 5 &= 2.2 + 1 \\ 2 &= 1.2 + 0 \\ 1 &= 0.2 + 1 \end{aligned}$$

$43_{10} = 2.(2.(2.(2.(2.(2.0 + 1) + 0) + 1) + 0) + 1) + 1$ donc $43_{10}$ s'écrit en binaire $101011_2$.

# *Méthode des soustractions successives*

*Rappel (corollaire 2) :* Pour tout entier $a$, il existe $k \in \mathbb{N}^*$ tel que $2^{k-1} \leq a < 2^k$).

*Principe :* Soustraire à $a$ la plus grande puissance de $b$ immédiatement inférieure à $a$, puis à répéter l'opération avec le résultat obtenu jusqu'à ce que le résultat de l'opération soit nul.

# *Algorithme*

- La puissance de $b$ inférieure à $a$ la plus proche (par exemple $b^m$) est soustraite de $a$, ce qui donne pour résultat $s_m$ ;

# Algorithme

- La puissance de $b$ inférieure à $a$ la plus proche (par exemple $b^m$) est soustraite de $a$, ce qui donne pour résultat $s_m$ ;
- si $s_m > 0$, la puissance de $b$ inférieure à $s_m$ la plus proche (par exemple $b^p, p < m$)est soustraite de $s_m$, ce qui donne pour résultat $s_p$ ;

# Algorithme

- La puissance de $b$ inférieure à $a$ la plus proche (par exemple $b^m$) est soustraite de $a$, ce qui donne pour résultat $s_m$ ;
- si $s_m > 0$, la puissance de $b$ inférieure à $s_m$ la plus proche (par exemple $b^p, p < m$)est soustraite de $s_m$, ce qui donne pour résultat $s_p$ ;
- l'opération de soustraction mentionnée ci-dessus est renouvelée sur chacun des $s_i$ obtenus ($0 \leq i \leq m$) jusqu'à obtenir un certain $i = z$ tel que $s_z = 0$ ;

# Algorithme

- La puissance de $b$ inférieure à $a$ la plus proche (par exemple $b^m$) est soustraite de $a$, ce qui donne pour résultat $s_m$ ;
- si $s_m > 0$, la puissance de $b$ inférieure à $s_m$ la plus proche (par exemple $b^p, p < m$)est soustraite de $s_m$, ce qui donne pour résultat $s_p$ ;
- l'opération de soustraction mentionnée ci-dessus est renouvelée sur chacun des $s_i$ obtenus ($0 \leq i \leq m$) jusqu'à obtenir un certain $i = z$ tel que $s_z = 0$ ;
- les restes obtenus par soustractions successives sont les chiffres de la représentation en base $b$ de l'entier $a$, avec $s_m$ comme chiffre le plus significatif, $s_p$ comme chiffre suivant, et ainsi de suite jusqu'à $s_z$ le chiffre le moins significatif.

# *Exemple*

Conversion du nombre $43_{10}$ en binaire :

$$\begin{array}{rcrcrcrcr} 2^5 & \leq & 43 & , & 43 & - & 2^5 & = & 11 \\ 2^3 & \leq & 11 & , & 11 & - & 2^3 & = & 3 \\ 2^1 & \leq & 3 & , & 3 & - & 2^1 & = & 1 \\ 2^0 & \leq & 1 & , & 1 & - & 2^0 & = & 0 \end{array}$$

$$\begin{aligned} 43 &= 2^5 + 2^3 + 2^1 + 2^0 \\ &= 1.2^5 + 0.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 \end{aligned}$$

autrement dit $43_{10}$ s'écrit en binaire $101011_2$.

# *Conversion binaire–hexadécimal*

Les $16$ chiffres du code hexadécimal se codent en binaire sur $4$ bits.

*Exemple :* $198_{10}$ s'écrit $11000110_2$ en binaire, $\underbrace{1100}_{C}$ $\underbrace{0110}_{6}$ et donc $C6_{16}$ en hexadécimal. $43_{10}$ s'écrit $101011_2$ en binaire, $\underbrace{0010}_{2}$ $\underbrace{1011}_{B}$ et donc $2B_{16}$ en hexadécimal.

# *0pérations binaires*

- Addition
- Soustraction
- Multiplication
- Division
- Décalage logique à gauche
- Décalage logique à droite

# Table d'addition binaire

$$
\begin{array}{lcl}
0+0 & = & 0 \\
0+1 & = & 1 \\
1+0 & = & 1 \\
1+1 & = & \underbrace{1}_{\textit{retenue}}\ 0
\end{array}
$$

*Exemple :*

$$
\begin{array}{cccccc}
 & & {}^{1}1 & {}^{1}1 & {}^{1}0 & 1 \\
+ & & & 1 & 1 & 1 \\
\hline
 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

# Table de soustraction binaire

$$
\begin{array}{lcrr}
0-0 & = & & 0 \\
1-0 & = & & 1 \\
1-1 & = & & 0 \\
0-1 & = & \underbrace{1}_{\textit{retenue}} & 1
\end{array}
$$

*Exemple :*

$$
\begin{array}{rcccc}
 & 1 & 0 & 1 & 1 \\
- & {\scriptstyle 1} & 1 & 0 & 1 \\
\hline
 & 0 & 1 & 1 & 0
\end{array}
$$

# Table de multiplication binaire

$$\begin{array}{|lcl|}\hline 0\times 0 & = & 0\\ 0\times 1 & = & 0\\ 1\times 0 & = & 0\\ 1\times 1 & = & 1\\ \hline\end{array}$$

*Exemple :*

$$\begin{array}{cccccccc} & & & & 1 & 0 & 1 & 1\\ \times & & & & & 1 & 0 & 1\\ \hline & & & & 1 & 0 & 1 & 1\\ & & & {}^{1}0 & 0 & 0 & 0 & \\ & & 1 & 0 & 1 & 1 & & \\ \hline & & 1 & 1 & 0 & 1 & 1 & 1 \end{array}$$

# Table de division binaire

| | | |
|---|---|---|
| $0 \div 0$ | : | indéfini |
| $0 \div 1$ | $=$ | $0$ |
| $1 \div 0$ | : | indéfini |
| $1 \div 1$ | $=$ | $1$ |

*Exemple :*

$$
\begin{array}{cccc|cc}
1 & 0 & 1 & 0 & 1 & 0 \\
\cline{5-6}
1 & 0 & & & 1 & 0 \; 1 \\
\cline{1-2}
0 & 0 & 1 & & & \\
 & 0 & 0 & & & \\
\cline{2-3}
 & 0 & 1 & 0 & & \\
 & & 1 & 0 & & \\
\cline{3-4}
 & & 0 & 0 & &
\end{array}
$$

# Décalages

$$\mathit{sll}(q_{n-1}\, q_{n-2} \ldots q_0) = q_{n-2} \ldots q_0\, 0$$
$$\mathit{slr}(q_{n-1}\, q_{n-2} \ldots q_0) = 0\, q_{n-1} \ldots q_1$$

# Décalages

$$\begin{aligned} \mathit{sll}(q_{n-1}\, q_{n-2} \ldots q_0) &= q_{n-2} \ldots q_0\, 0 \\ \mathit{slr}(q_{n-1}\, q_{n-2} \ldots q_0) &= 0\, q_{n-1} \ldots q_1 \end{aligned}$$

**Théorème 4** *Etant donnée* $q_{n-1}\, q_{n-2} \ldots q_0$ *la représentation sur* $n$ *bits d'un entier naturel* $a$*, on a :*

$$\begin{aligned} \mathit{sll}(q_{n-1}\, q_{n-2} \ldots q_0) &= 2 \times a \\ \mathit{slr}(q_{n-1}\, q_{n-2} \ldots q_0) &= a \div 2 \end{aligned}$$

# *Dépassement de capacité*

- Une retenue au delà du bit de poids fort produit un dépassement de capacité : sur quatre bits $7 + 12 = 19$ qui ne peut être représenté

# *Dépassement de capacité*

- Une retenue au delà du bit de poids fort produit un dépassement de capacité : sur quatre bits $7 + 12 = 19$ qui ne peut être représenté
- La détection d'un dépassement de capacité est immédiate :

$$a_2 + b_2 = a_2 \oplus b_2 + (R.2^n)_2 \quad \text{où } R \in \{0, 1\}$$

Si dépassement $R = 1$, sinon $R = 0$

# *Représentation des nombres signés*

- Sur $n$ bits : $2^n$ configurations binaires

# *Représentation des nombres signés*

- Sur $n$ bits : $2^n$ configurations binaires
- Entiers signés $\rightarrow$ diviser cet ensemble de configurations par $2$

# *Représentation des nombres signés*

- Sur $n$ bits : $2^n$ configurations binaires
- Entiers signés $\rightarrow$ diviser cet ensemble de configurations par $2$
- Une moitié des configurations pour les entiers positifs, l'autre moitié pour les entiers négatifs :

$$\underbrace{-2^{n-1} \longleftrightarrow 0 \longleftrightarrow 2^{n-1} - 1}_{2^n \textit{ valeurs distinctes}}$$

# *Représentation des nombres signés*

- Sur $n$ bits : $2^n$ configurations binaires
- Entiers signés $\rightarrow$ diviser cet ensemble de configurations par $2$
- Une moitié des configurations pour les entiers positifs, l'autre moitié pour les entiers négatifs :

$$\underbrace{-2^{n-1} \longleftrightarrow 0 \longleftrightarrow 2^{n-1} - 1}_{2^n \textit{ valeurs distinctes}}$$

- Donc $n - 1$ bits pour les entiers positifs, comme les entiers négatifs

# *Bit de signe*

La distinction sur le signe découle du positionnement à $1$ ou $0$ du $n^{\text{ème}}$ bit de poids fort, appelé bit de signe :

$$\underbrace{q^{n-1}}_{\textit{bit de signe}} \qquad \underbrace{q^{n-2} \ldots q^1 \; q^0}_{\textit{valeur absolue } (n-1 \textit{ bits})}$$

Bit de signe $= 0 \rightarrow$ entiers positifs
Bit de signe $= 1 \rightarrow$ entiers négatifs

# *Codage des entiers*

- codage en signe et valeur absolue
- codage en complément à un
- codage en complément à deux

# *Signe et valeur absolue*

Distinction assurée seulement par le bit de signe

$$+1 \quad \rightarrow \quad 0001$$
$$-1 \quad \rightarrow \quad 1001$$

# *Signe et valeur absolue*

Distinction assurée seulement par le bit de signe

$$+1 \quad \rightarrow \quad 0001$$
$$-1 \quad \rightarrow \quad 1001$$

Problème : Deux représentations de zéro. . .

$$0^{+} \quad \rightarrow \quad 0000$$
$$0^{-} \quad \rightarrow \quad 1000$$

# *Complément à un*

On inverse les $1$ et les $0$

$$\begin{array}{lcl} +1 & \rightarrow & 0001 \\ -1 & \rightarrow & 1110 \end{array}$$

# *Complément à un*

On inverse les $1$ et les $0$

$$\begin{array}{lcl} +1 & \rightarrow & 0001 \\ -1 & \rightarrow & 1110 \end{array}$$

Problème : Deux représentations de zéro. . .

$$\begin{array}{lcl} 0^{+} & \rightarrow & 0000 \\ 0^{-} & \rightarrow & 1111 \end{array}$$

# *Complément à deux*

Un entier négatif est tel qu'additionné à l'entier positif correspondant, on obtienne bien un zéro unique

$$\begin{array}{rcccc} & 0 & 0 & 0 & 1 \\ - & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 \end{array}$$

Codage de $-1$ :

$$\begin{array}{rcccc} & 0 & 0 & 0 & 0 \\ - & {}^{1}0 & {}^{1}0 & {}^{1}0 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}$$

# *Dépassement de capacité*

- La génération d'une retenue au-delà du bit de poids le plus fort permet de diviser l'ensemble des configurations binaires en deux et permet de constituer le bit de signe

# *Dépassement de capacité*

- La génération d'une retenue au-delà du bit de poids le plus fort permet de diviser l'ensemble des configurations binaires en deux et permet de constituer le bit de signe
- Si on ignore le dépassement de capacité, la représentation obtenue est compatible avec les comportements attendus de l'addition et de la soustraction

# *Dépassement de capacité*

- La génération d'une retenue au-delà du bit de poids le plus fort permet de diviser l'ensemble des configurations binaires en deux et permet de constituer le bit de signe
- Si on ignore le dépassement de capacité, la représentation obtenue est compatible avec les comportements attendus de l'addition et de la soustraction
- Ignorer un dépassement de capacité sur le $n+1^{\text{ème}}$ bit revient à considérer un codage des entiers modulo $2^n$

# *Rappel : congruences*

**Définition 1** *Etant donnés $x \in \mathbb{Z}$, $y \in \mathbb{Z}$, et $n \in \mathbb{N}^*$, $x$ est dit congru à $y$ modulo $n$ (noté $x \equiv y \ (\text{mod}\ n)$) si et seulement s'il existe un entier rationnel $k$ tel que $x = y + k.n$.*

**Théorème 5** *Soient $x \in \mathbb{Z}$, $y \in \mathbb{Z}$, $n \in \mathbb{N}^*$, $x \equiv y \ (\text{mod}\ n)$ ssi $x$ et $y$ ont le même reste dans la division euclidienne par $n$.*

# *Congruences (suite)*

**Définition 2** *Etant donnés* $x \in \mathbb{Z}$, *et* $n \in \mathbb{N}^*$,

$$x \textit{ modulo } n = y \quad \textit{ssi} \quad x = n.k + y, \textit{avec } r \in \mathbb{Z}, k \in \mathbb{Z}$$

*Remarque :* Compatible avec l'addition et la multiplication dans $\mathbb{Z}$. $\forall (x, x') \in \mathbb{Z} \times \mathbb{Z}$, $\forall (y, y') \in \mathbb{Z} \times \mathbb{Z}$,

$$\text{si } \begin{cases} x \equiv x' \pmod{n} \\ y \equiv y' \pmod{n} \end{cases}$$

alors $x + y \equiv x' + y' \pmod{n}$ et $x.y \equiv x'.y' \pmod{n}$

# *Classes d'équivalence modulo* $n$

- La relation de congruence modulo $n$ est une relation d'équivalence. L'ensemble des classes d'équivalence est noté $\mathbb{Z}/n\mathbb{Z}$

# *Classes d'équivalence modulo* $n$

- La relation de congruence modulo $n$ est une relation d'équivalence. L'ensemble des classes d'équivalence est noté $\mathbb{Z}/n\mathbb{Z}$
- Chaque classe d'équivalence est caractérisée par un des restes possibles de la division euclidienne par $n$ : $0, 1, \ldots n-1$. Par exemple, pour $n = 4$, on dispose de la classe d'équivalence $\hat{0} = \{\ldots, -8, -4, 0, 4, 8, 12, \ldots\}$

# *Classes d'équivalence modulo* $n$

- La relation de congruence modulo $n$ est une relation d'équivalence. L'ensemble des classes d'équivalence est noté $\mathbb{Z}/n\mathbb{Z}$
- Chaque classe d'équivalence est caractérisée par un des restes possibles de la division euclidienne par $n$ : $0, 1, \ldots n-1$. Par exemple, pour $n = 4$, on dispose de la classe d'équivalence $\hat{0} = \{\ldots, -8, -4, 0, 4, 8, 12, \ldots\}$
- $\mathbb{Z}/n\mathbb{Z} = \{\hat{0}, \hat{1}, \ldots \hat{n-1}\}$

# *Exemple :* $\mathbb{Z}/4\mathbb{Z}$

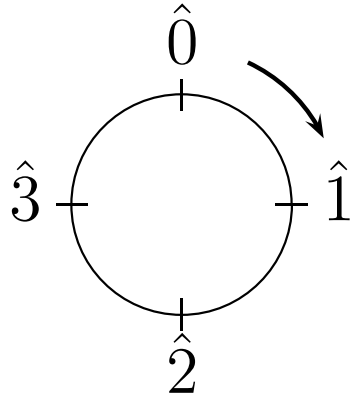
Les éléments de $\mathbb{Z}/4\mathbb{Z}$ sont :

$$\begin{aligned}
\hat{0} &= \{\ldots, -8, -4, 0, 4, 8, 12, \ldots\} \\
\hat{1} &= \{\ldots, -7, -3, 1, 5, 9, 13, \ldots\} \\
\hat{2} &= \{\ldots, -6, -2, 2, 6, 10, 14, \ldots\} \\
\hat{3} &= \{\ldots, -5, -1, 3, 7, 11, 15, \ldots\}
\end{aligned}$$

*Remarque :* $\hat{0}$ dans $\mathbb{Z}/4\mathbb{Z}$ est différent de $\hat{0}$ dans $\mathbb{Z}/5\mathbb{Z}$ :

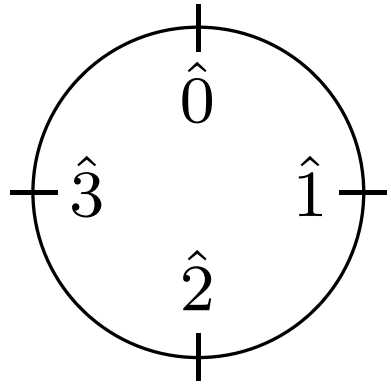
$$\begin{aligned}
\hat{0} &= \{\ldots, -8, -4, 0, 4, 8, \ldots\} && \text{dans } \mathbb{Z}/4\mathbb{Z} \\
\hat{0} &= \{\ldots, -10, -5, 0, 5, 10, \ldots\} && \text{dans } \mathbb{Z}/5\mathbb{Z}
\end{aligned}$$

# *Représentation sur un cercle*

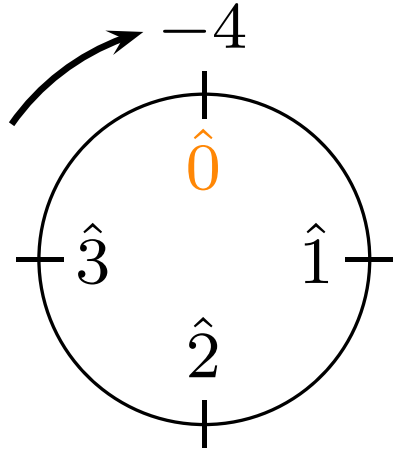



La position des différents éléments de chaque classe d'équivalence sur le cercle est matérialisée par le nombre de tours successifs réalisés sur le cercle.

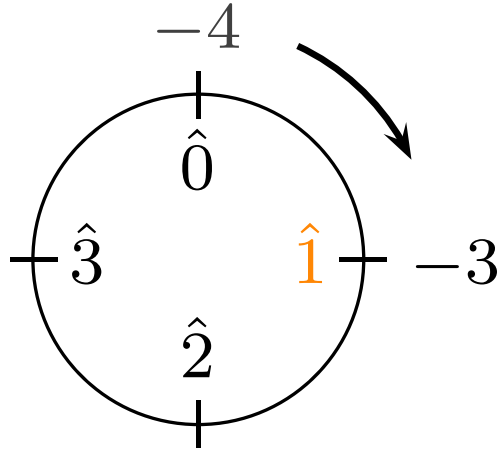
# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$

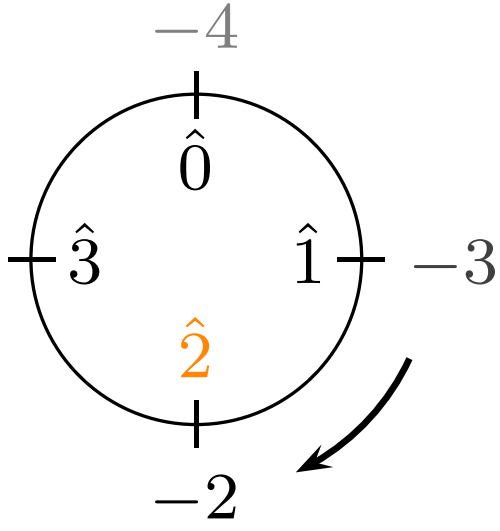


# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



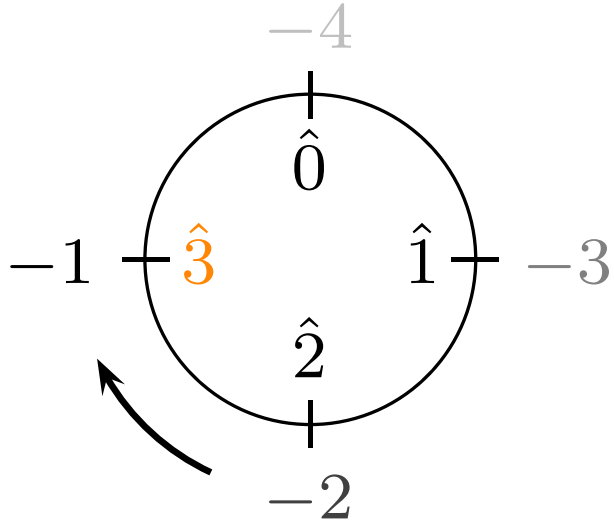

Tour 1

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



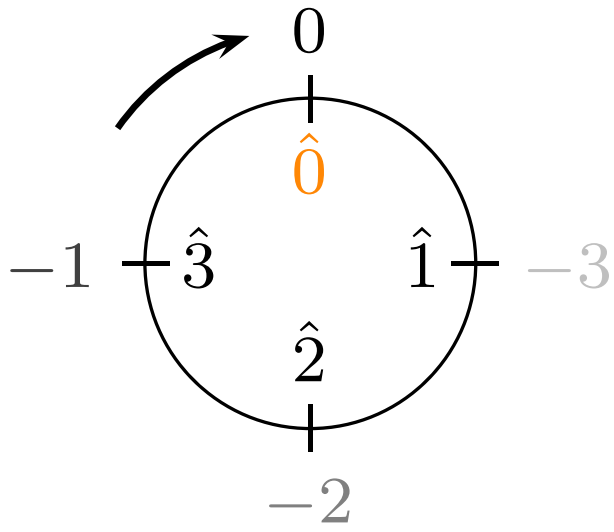

Tour 1

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



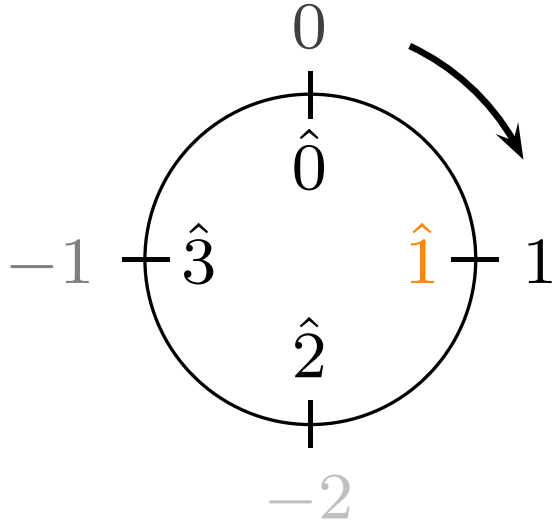

Tour 1

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



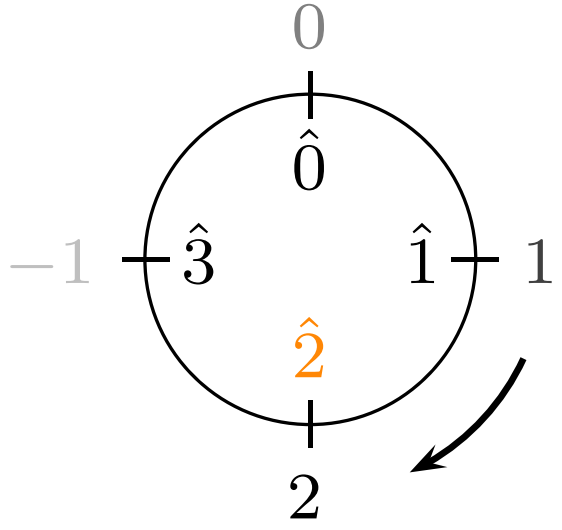

Tour 1

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



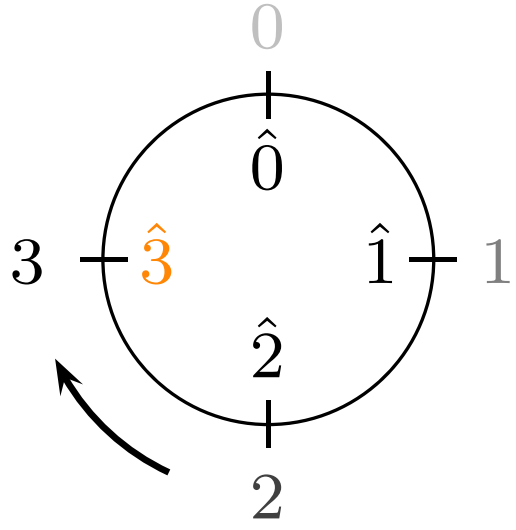

Tour 2

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



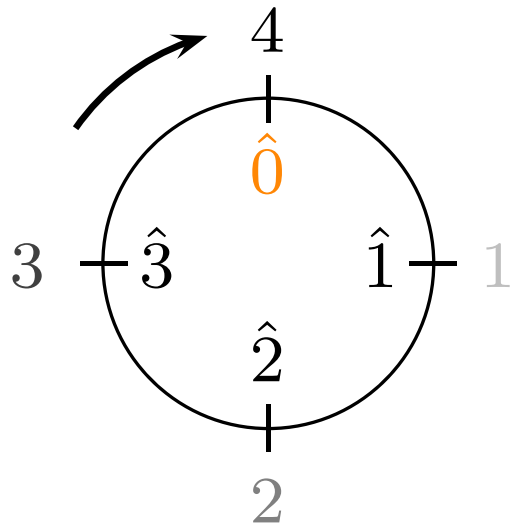

Tour 2

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



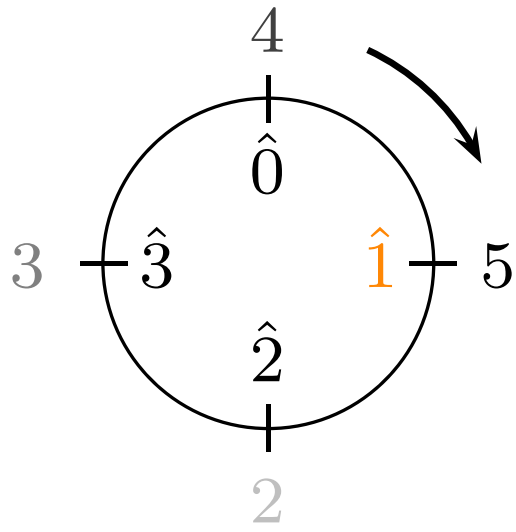

Tour 2

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



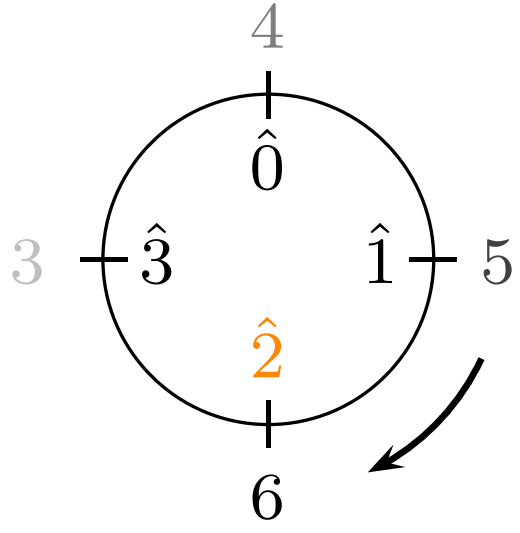

Tour 2

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



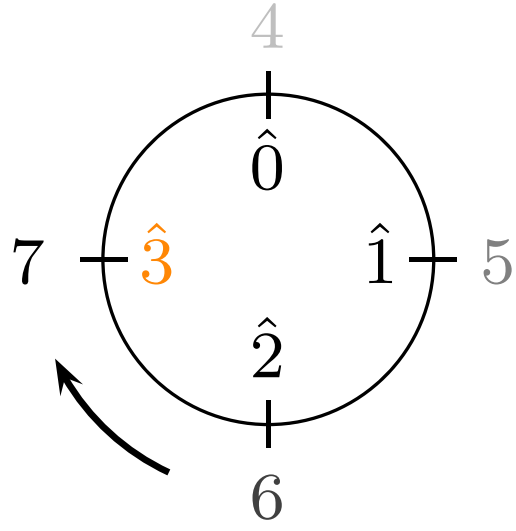

Tour 3

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$



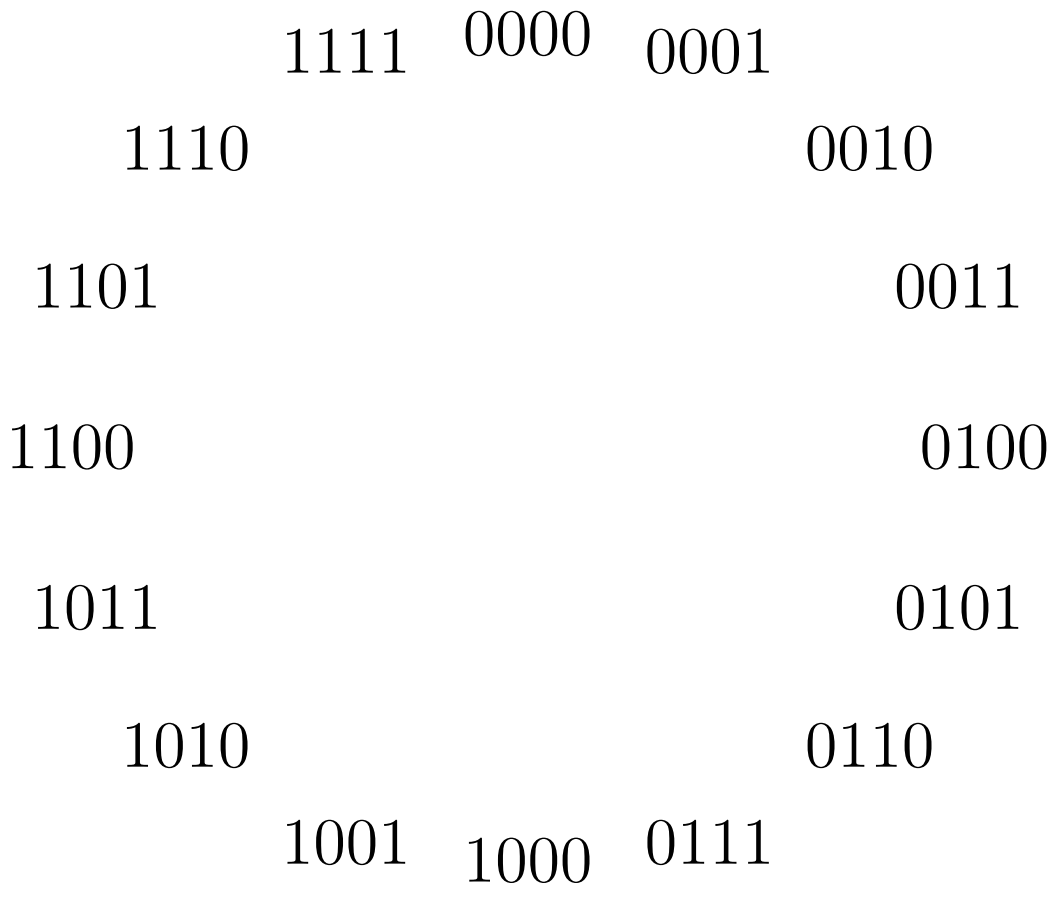

Tour 3

# *Exemple (suite) :* $\mathbb{Z}/4\mathbb{Z}$

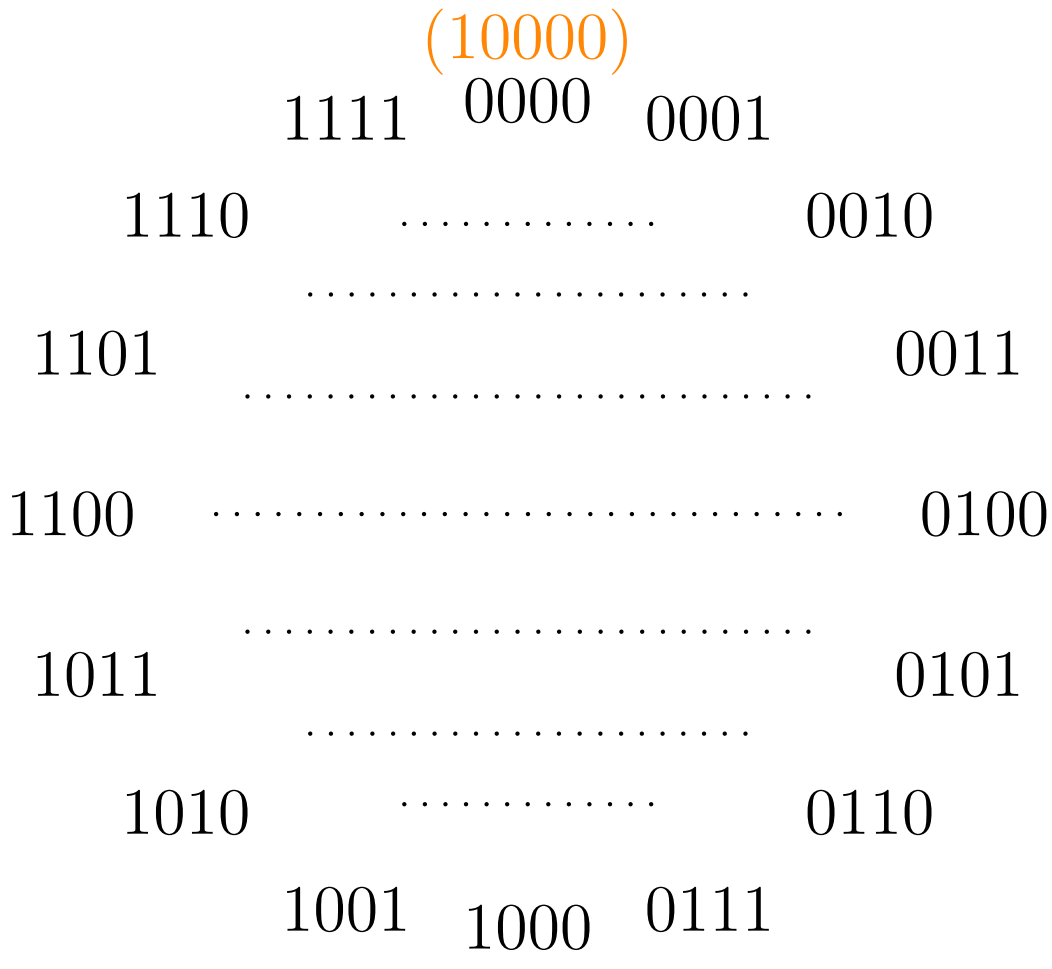



Tour 3

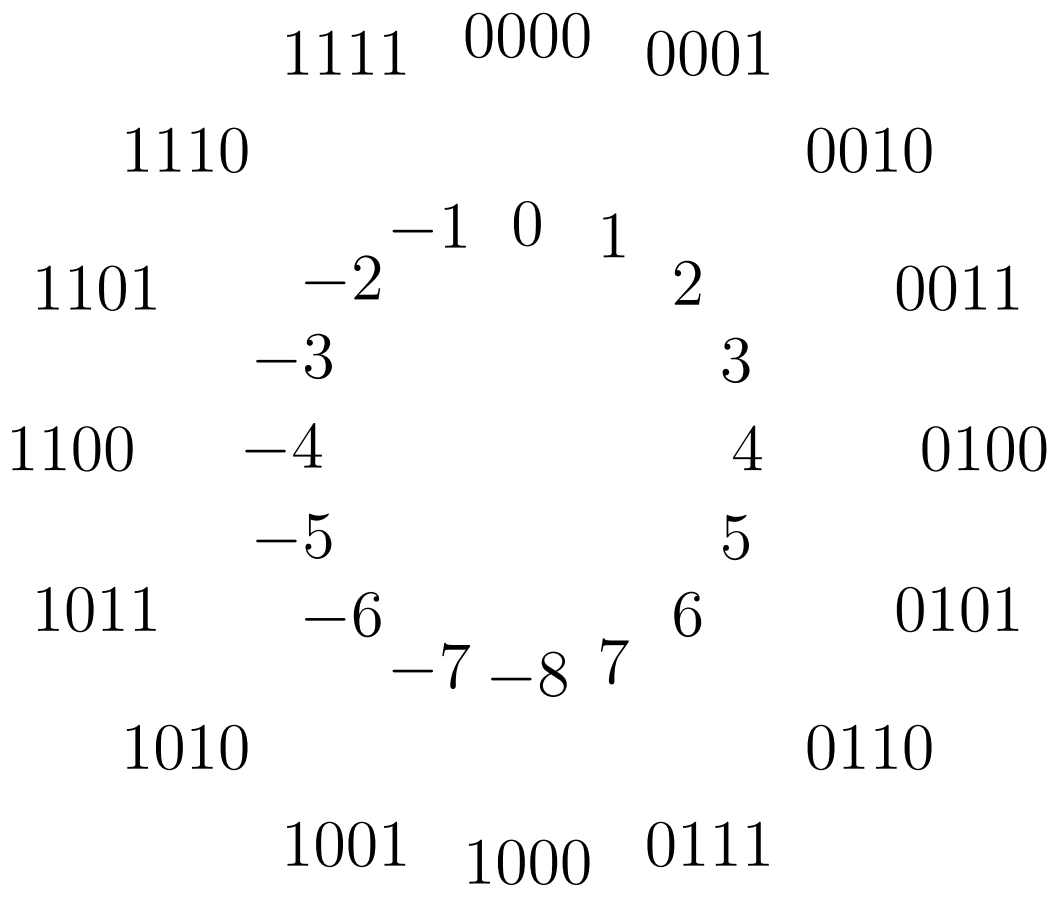
# Complément à deux sur quatre bits



# Complément à deux sur quatre bits



# Complément à deux sur quatre bits



# *Calcul du complément à deux*

**Théorème 6** *Etant donné $a \in [-2^{n-1}+1, 2^{n-1}-1]$, la représentation de $-a$ en complément à $2$ est obtenue à partir du complément à $1$ de $a$ auquel on ajoute $1$.*

*Exemple :*

| | | | | | |
|---|---|---|---|---|---|
| a | : | | $0$ | $0$ | $0$ | $1$ |
| Complément à $1$ | : | | $1$ | $1$ | $1$ | $0$ |
| ajoût de $1$ | : | $+$ | | | | $1$ |
| -a | : | | $1$ | $1$ | $1$ | $1$ |

# *Extension de signe*

**Théorème 7** *Etant donné un entier relatif $a$ représenté sur $n$ bits par $q_{n-1}\, q_{n-2} \ldots q_0$, $a$ est représenté sur $m$ $(m > n)$ bits par $q_{n-1} \ldots q_{n-1}\, q_{n-2} \ldots q_0$.*

*Exemple :*

$+7 \quad \rightarrow \quad 0111$ sur quatre bits, $00000111$ sur un octet

$-2 \quad \rightarrow \quad 1110$ sur quatre bits, $11111110$ sur un octet.

# *Dépassement de capacité*

**Théorème 8** *L'addition UAL de deux entiers relatifs $a$ et $b$ fournit toujours un résultat correct :*

- *si $a$ et $b$ ne sont pas de mêmes signes*
- *si $a$ et $b$ sont de même signe, quand le bit de signe est égal à la retenue*

*Exemple :*

$$\begin{array}{rcl} 0111 & \rightarrow & 7 \\ + \quad 0010 & \rightarrow & 2 \\ \hline 1001 & \rightarrow & -7 \end{array}$$