Langage C avancé
Représentation des données en mémoire

Samuel KOKH

samuel.kokh@cea.fr

MACS 1 – Institut Galilée

Vers un aperçu de la gestion de la mémoire à bas niveau

- comprendre représenter des « objets » par des 0 et des 1
- mieux comprendre la « philosophie » du langage
- approche plus naturelle de la mécanique des pointeurs

Le Bit

Qu'est-ce qu'un bit ?

BIT = abbréviation de « Binary digIT »

C'est la plus petite unité d'information stockée et compréhensible par la machine.

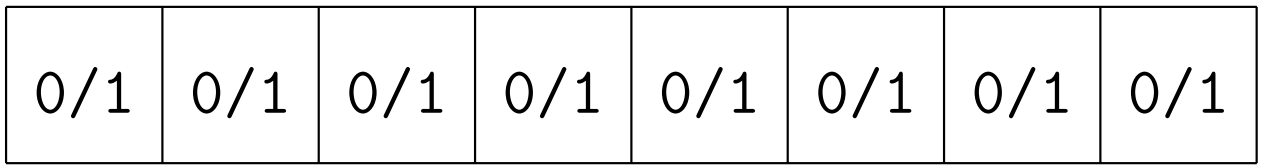
Représentation d'un bit

- Symboliquement : variable $a = 0/1$
- Symboliquement : case dont le contenu ne peut être que $0/1$ ou vrai/faux
- low/high voltage

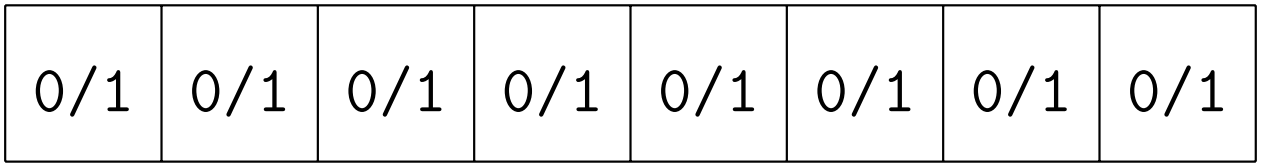
Octet/Byte

Les bits sont habituellement manipulés par groupe de 8 bits.

Un groupe de 8 bits = un octet (français) / un byte (anglais)



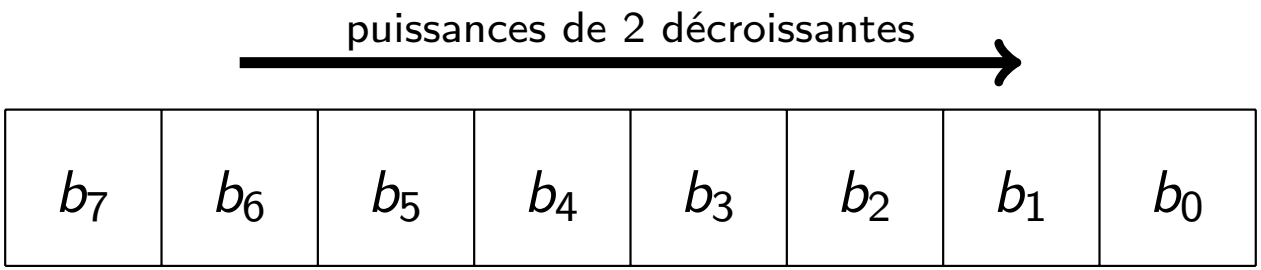
= un octet



Un octet peut prendre $2^8 = 256$ valeurs différentes

C'est le nombre de caractères qu'il y a dans la table ASCII.

Convention de lecture des bits (pour la suite des diapos)



représente l'entier

$$a_7 \times 2^7 + a_6 \times 2^6 + a_5 \times 2^5 + a_4 \times 2^4 + a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$$

ASCII : mise en concordance des valeurs d'un octet avec un signe/caractère « visible » à l'écran.

0	1	0	0	0	0	0	1

$\sim 65 \quad \mapsto \quad \text{CHAR}_{65} \sim \text{CHAR}_{01000001} \sim \text{'A'}$

ASCII : mise en concordance des valeurs d'un octet avec un signe/caractère « visible » à l'écran.

ASCII : mise en concordance des valeurs d'un octet avec un signe/caractère « visible » à l'écran.

0	1	0	0	0	0	0	1

$\sim 65 \quad \mapsto \quad \text{CHAR}_{65} \sim \text{CHAR}_{01000001} \sim \texttt{'A'}$

$$2^6 + 2^0 = 64 + 1 = 65$$

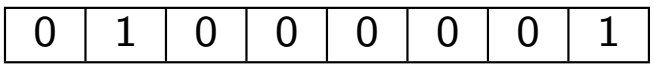
ASCII : mise en concordance des valeurs d'un octet avec un signe/caractère « visible » à l'écran.

0	1	0	0	0	0	0	1

$\sim 65 \quad \mapsto \quad \text{CHAR}_{65} \sim \text{CHAR}_{01000001} \sim$ 'A'

$$2^6 + 2^0 = 64 + 1 = 65$$

ASCII : mise en concordance des valeurs d'un octet avec un signe/caractère « visible » à l'écran.



$\sim 65 \quad \mapsto \quad \text{CHAR}_{65} \sim \text{CHAR}_{01000001} \sim$ 'A'

$$2^6 + 2^0 = 64 + 1 = 65$$

ASCII Table

La table ASCII avec une indexation décimale.

000	(nul)	016 ▶	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◀	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 –	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 ∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Les types d'entiers

Plusieurs type d'entiers

- (unsigned) char
- (unsigned) short
- (unsigned) int
- (unsigned) long int

Par convention (norme)

char stocké sur 1 octet

On supposera par la suite que (pas de norme : dépend des compilateurs/machines)

- short : stocké sur 2 octets
- int : stocké sur 4 octets

Les types d'entiers

Plusieurs type d'entiers

- (unsigned) char
- (unsigned) short
- (unsigned) int
- (unsigned) long int

Par convention (norme)

char stocké sur 1 octet

On supposera par la suite que (pas de norme : dépend des compilateurs/machines)

- short : stocké sur 2 octets
- int : stocké sur 4 octets

Les types d'entiers

Plusieurs type d'entiers

- (unsigned) char
- (unsigned) short
- (unsigned) int
- (unsigned) long int

Par convention (norme)

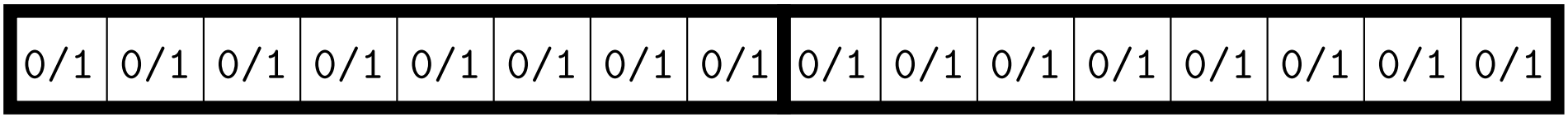
char stocké sur 1 octet

On supposera par la suite que (pas de norme : dépend des compilateurs/machines)

- short : stocké sur 2 octets
- int : stocké sur 4 octets

Représentation et opérations sur des entiers

On considère des entiers stockés sur 2 octets : (unsigned) short.



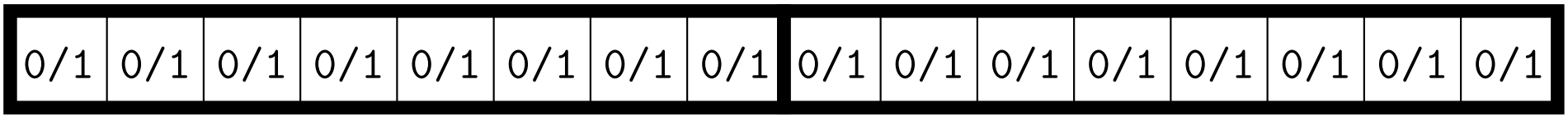
2^8 valeurs 2^8 valeurs

On peut donc représenter $2^{16} = 65536$ entiers différents avec 2 octets.

Par exemple, on peut représenter tous les nombres entiers de 0 à 65535

Représentation et opérations sur des entiers

On considère des entiers stockés sur 2 octets : (unsigned) short.



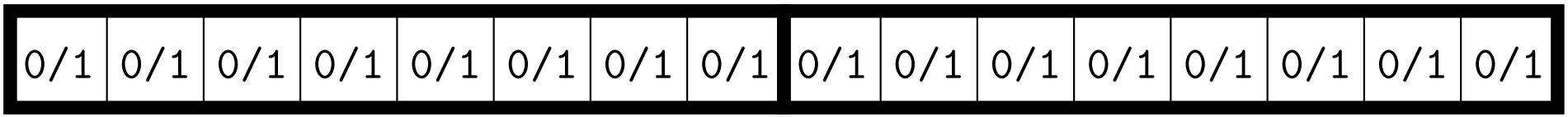
2^8 valeurs 2^8 valeurs

On peut donc représenter $2^{16} = 65536$ entiers différents avec 2 octets.

Par exemple, on peut représenter tous les nombres entiers de 0 à 65535

Représentation et opérations sur des entiers

On considère des entiers stockés sur 2 octets : (unsigned) short.



2^8 valeurs 2^8 valeurs

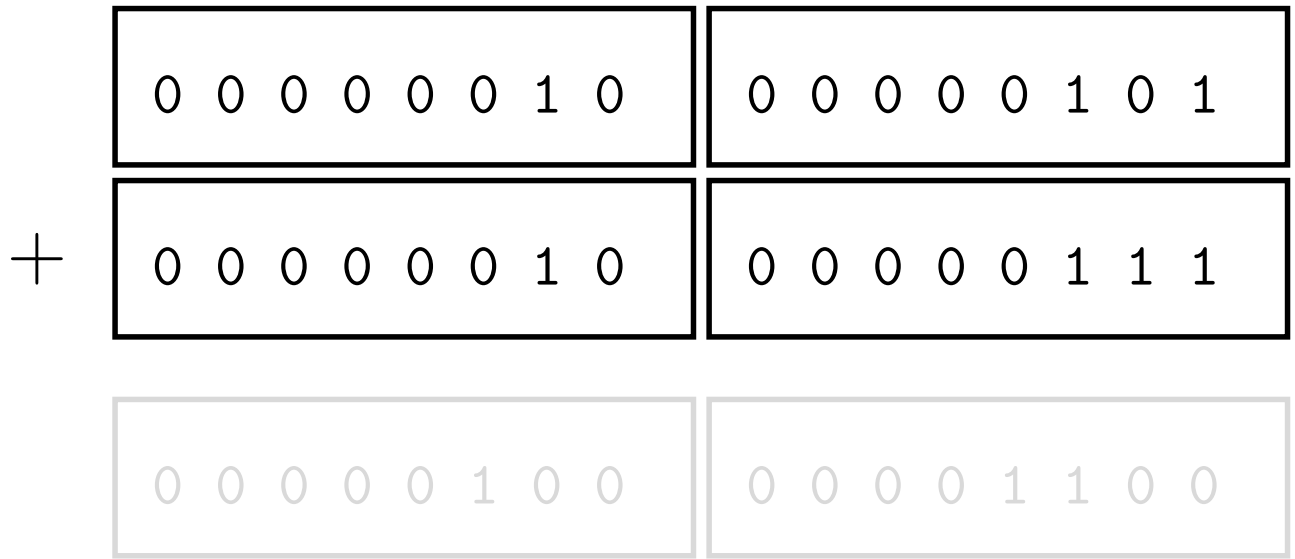
On peut donc représenter $2^{16} = 65536$ entiers différents avec 2 octets.

Par exemple, on peut représenter tous les nombres entiers de 0 à 65535

Addition sur 2 octets

Comment faire une addition ?

$$517 + 519 = ?$$

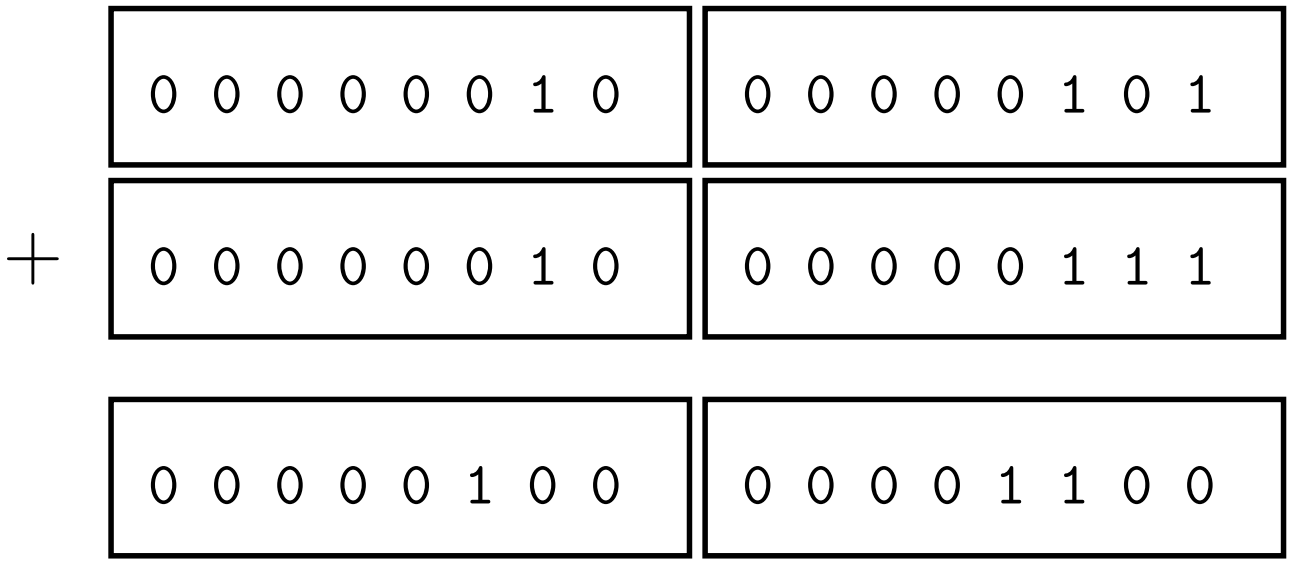


$$2^{10} + 2^3 + 2^2 = 1024 + 8 + 4 = 1036$$

Addition sur 2 octets

Comment faire une addition ?

$$517 + 519 = ?$$



$$2^{10} + 2^3 + 2^2 = 1024 + 8 + 4 = 1036$$

Addition sur 2 octets

Comment faire une addition ?

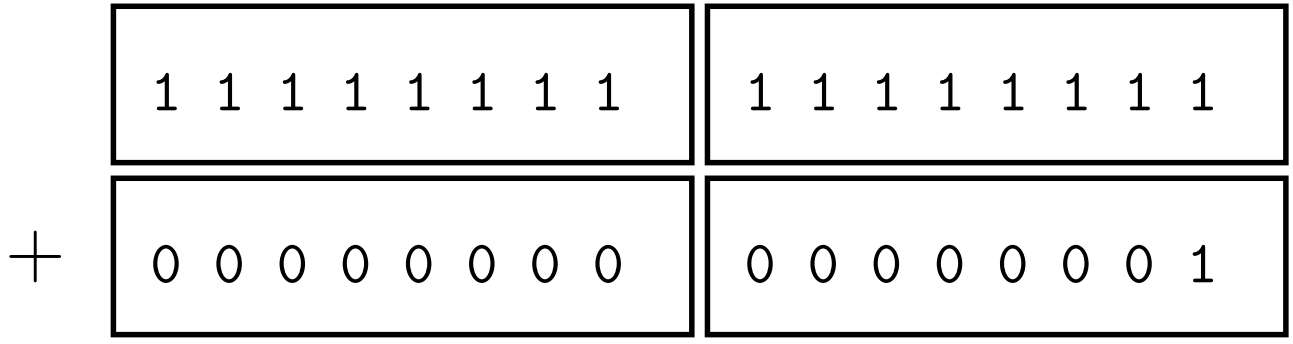
$$517 + 519 = ?$$

	0 0 0 0 0 0 1 0	0 0 0 0 0 1 0 1
+	0 0 0 0 0 0 1 0	0 0 0 0 0 1 1 1
	0 0 0 0 0 1 0 0	0 0 0 0 1 1 0 0

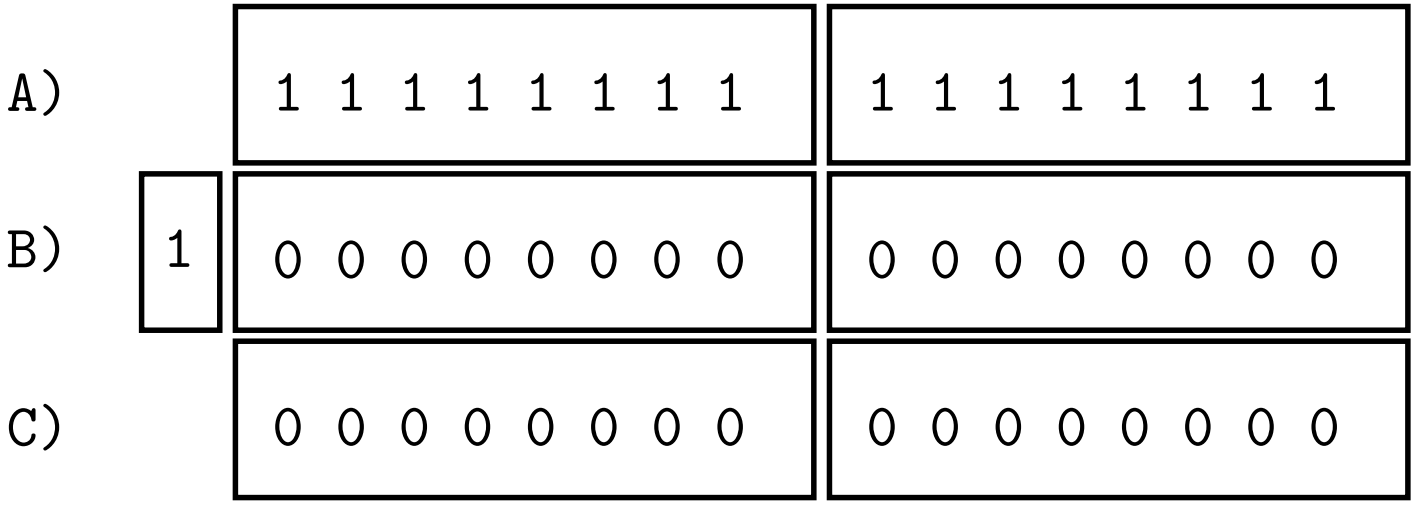
$$2^{10} + 2^3 + 2^2 = 1024 + 8 + 4 = 1036$$

Problème

Que se passe-t'il si on cherche à calculer



A)



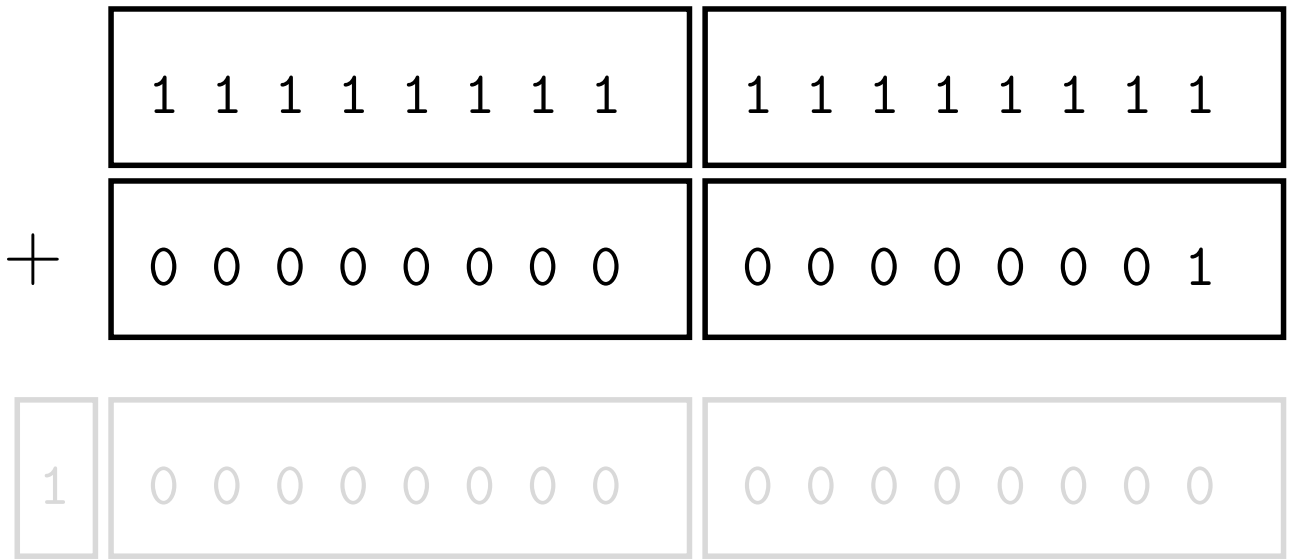
B)

C)

D) Ni A), ni B), ni C)

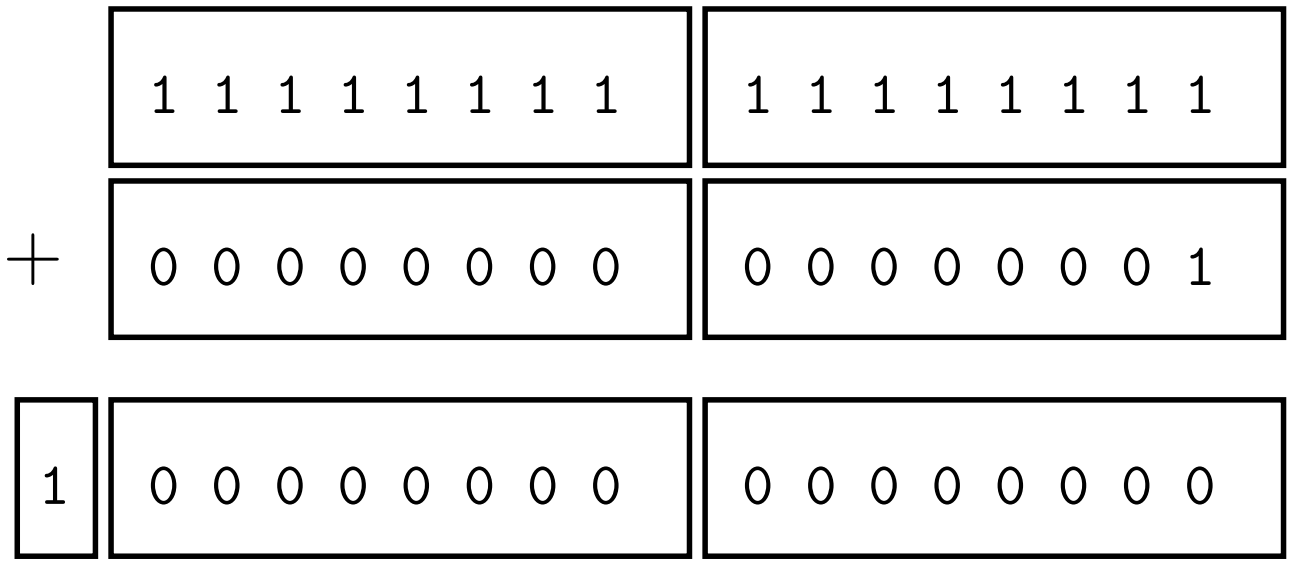
Problème

Que se passe-t'il si on cherche à calculer



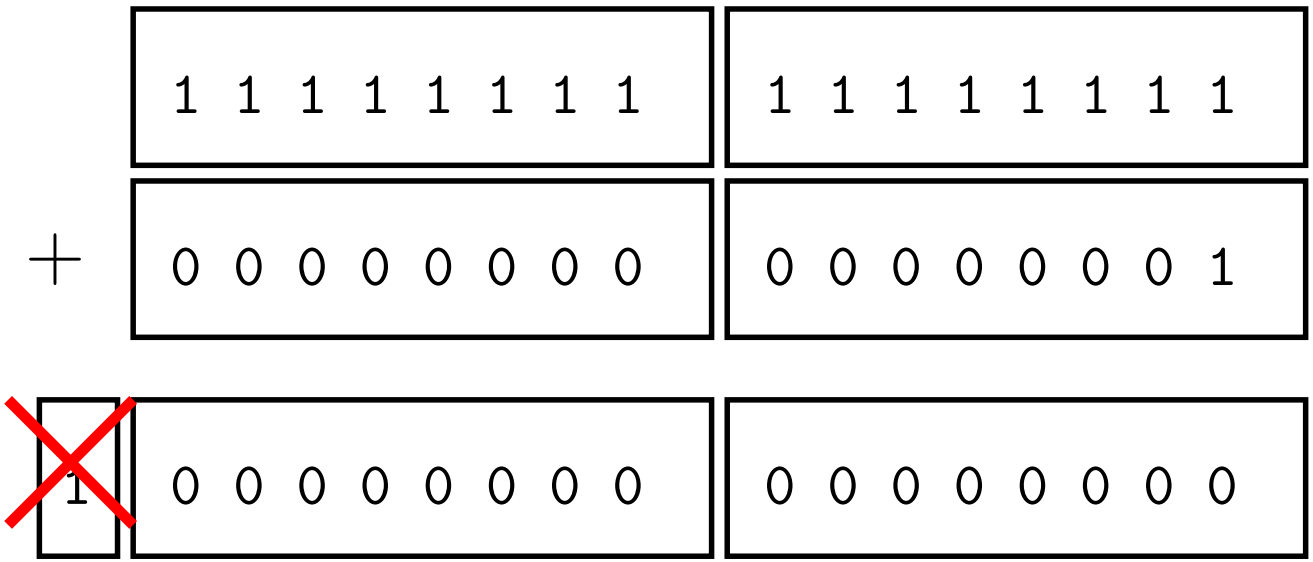
Problème

Que se passe-t'il si on cherche à calculer

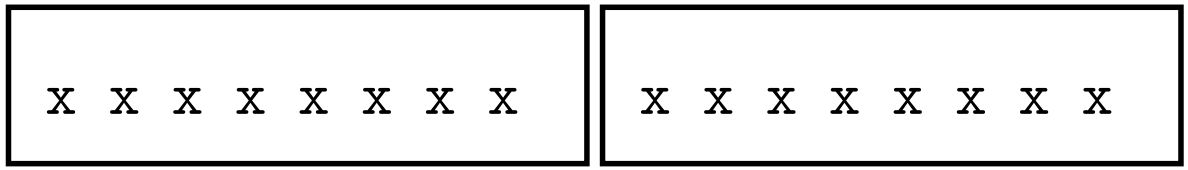


Problème

Que se passe-t'il si on cherche à calculer



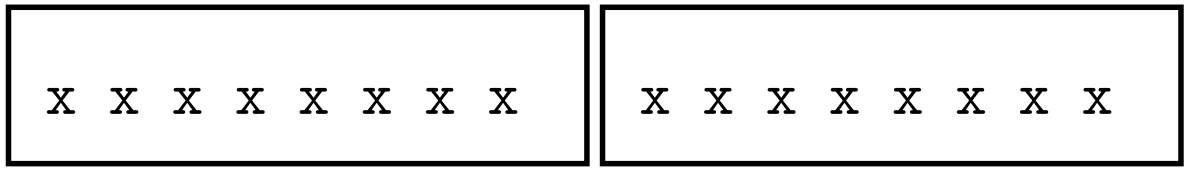
Comment représenter des entiers négatifs (sur 2 octets) ?

Soit 2 octets 

- si on veut représenter les nombres de 0 à $(2^{16} - 1)$: on a épuisé toutes les réprésentations possibles
- on doit « couper la poire en deux » : moitié de positifs, moitié de négatifs

Ajout d'une information sur le signe quelque part dans les deux octets.

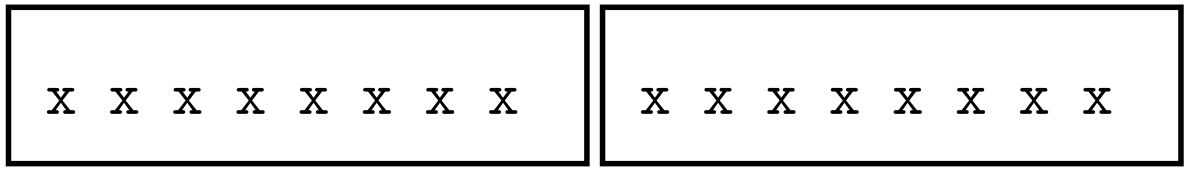
Comment représenter des entiers négatifs (sur 2 octets) ?

Soit 2 octets 

- si on veut représenter les nombres de 0 à $(2^{16} - 1)$: on a épuisé toutes les réprésentations possibles
- on doit « couper la poire en deux » : moitié de positifs, moitié de négatifs

Ajout d'une information sur le signe quelque part dans les deux octets.

Comment représenter des entiers négatifs (sur 2 octets) ?

Soit 2 octets 

- si on veut représenter les nombres de 0 à $(2^{16} - 1)$: on a épuisé toutes les réprésentations possibles
- on doit « couper la poire en deux » : moitié de positifs, moitié de négatifs

Ajout d'une information sur le signe quelque part dans les deux octets.

Essai

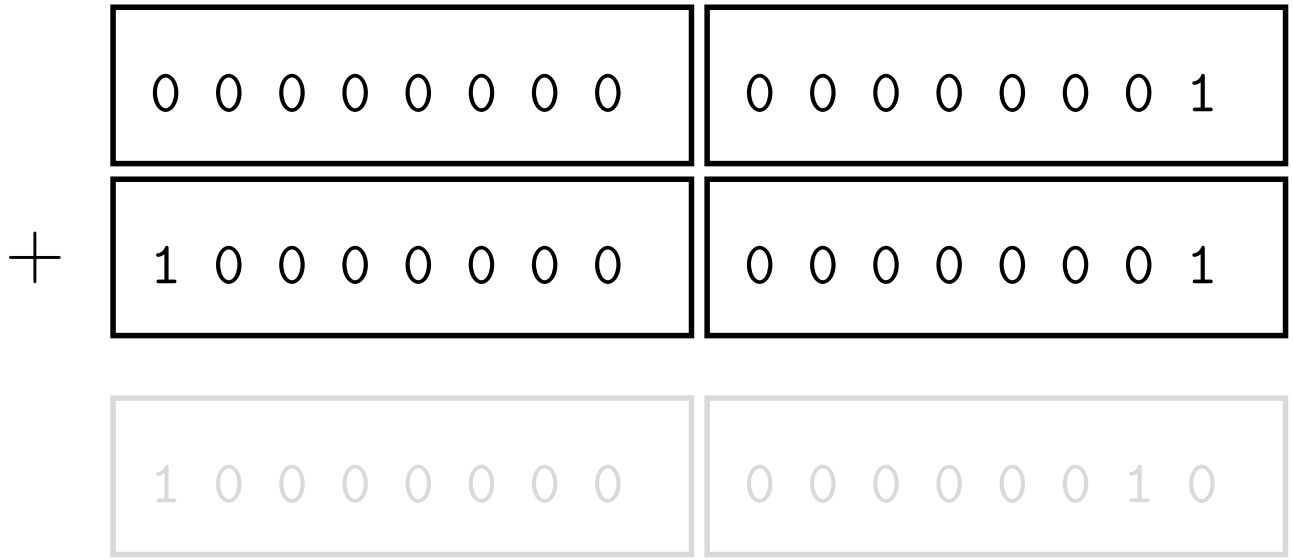
Que se passe-t'il si on réserve le « premier » bit pour désigner le signe ?

- description de 0
- description de $(2^{15} - 1)$ nombres strictements positifs
- description de 2^{15} nombres strictements négatifs

Essai

Que se passe-t'il si on réserve le « premier » bit pour désigner le signe ?

- description de 0
- description de $(2^{15} - 1)$ nombres strictements positifs
- description de 2^{15} nombres strictements négatifs

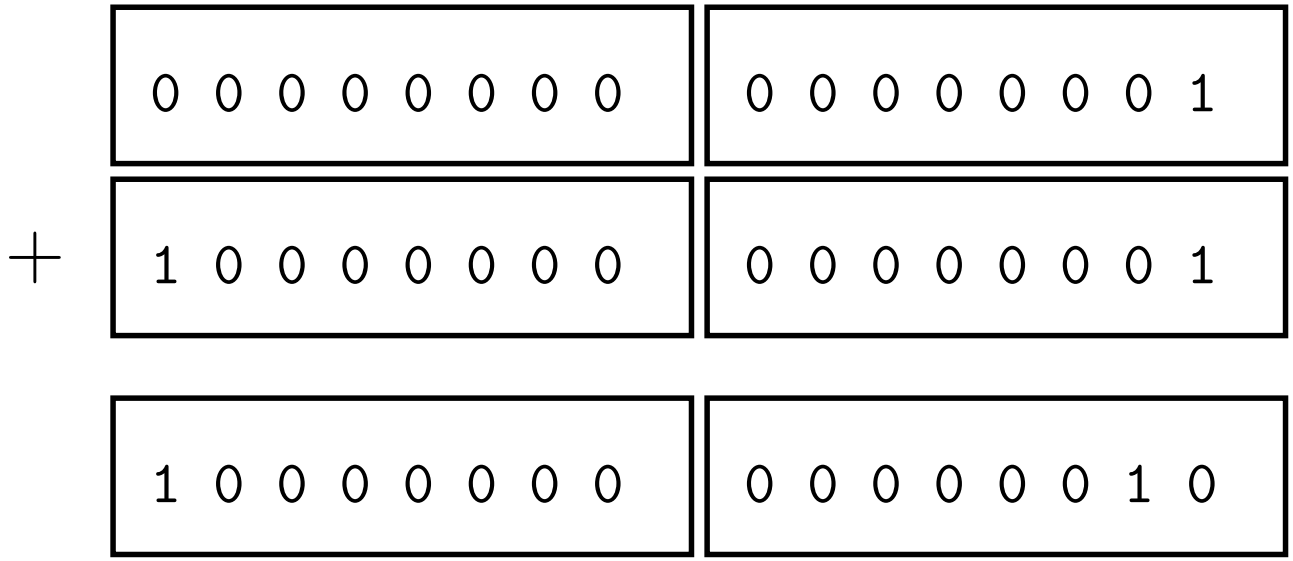


Avec la régle de signe précédente : quelle opération est représentée ici ?

a) $1 + (-1) = 0$

b) $1 + (-3) = -2$

c) ni l'un, ni l'autre

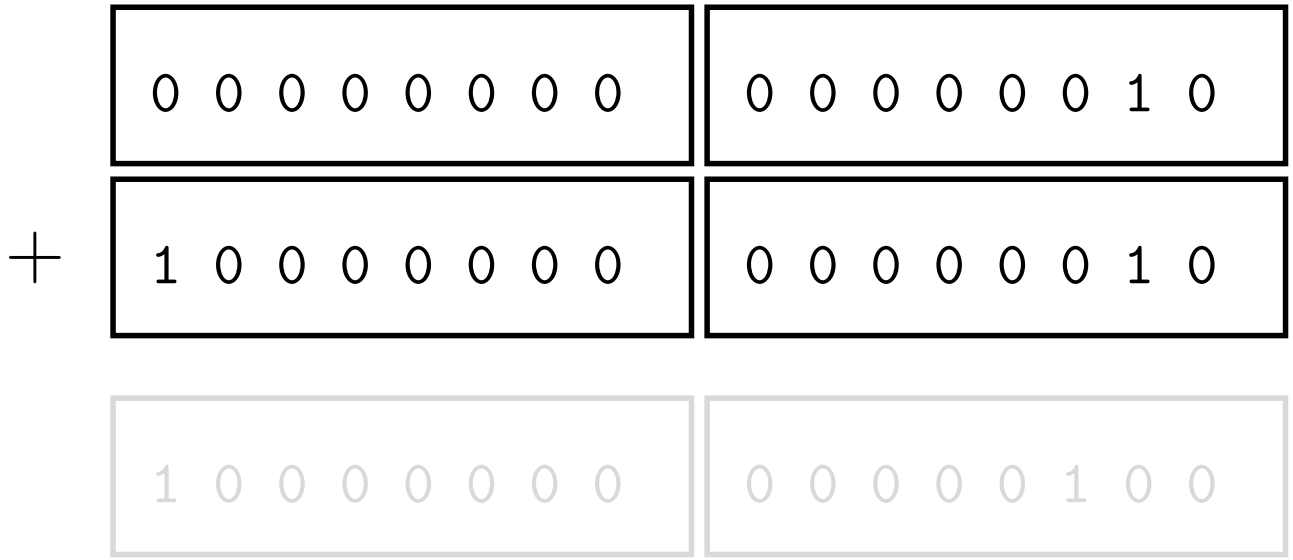


Avec la régle de signe précédente : quelle opération est représentée ici ?

a) $1 + (-1) = 0$

b) $1 + (-3) = -2$

c) ni l'un, ni l'autre

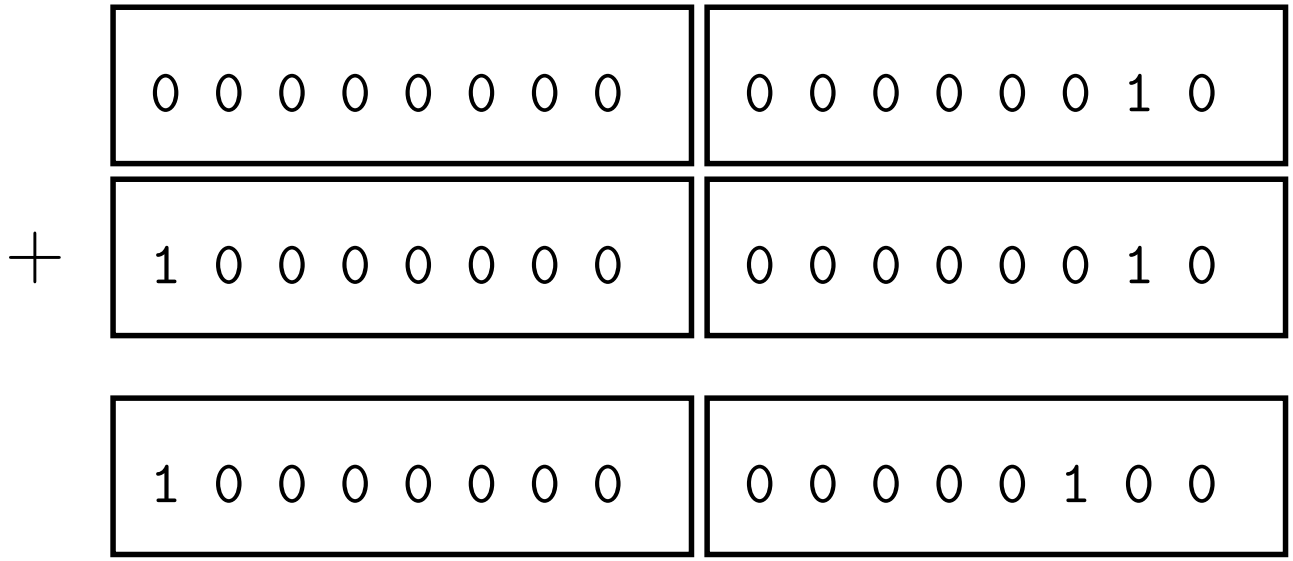


Avec la régle de signe précédente : quelle opération est représentée ici ?

a) $2 + (-2) = 0$

b) $2 + (-6) = -4$

c) ni l'un, ni l'autre



Avec la régle de signe précédente : quelle opération est représentée ici ?

a) $2 + (-2) = 0$

b) $2 + (-6) = -4$

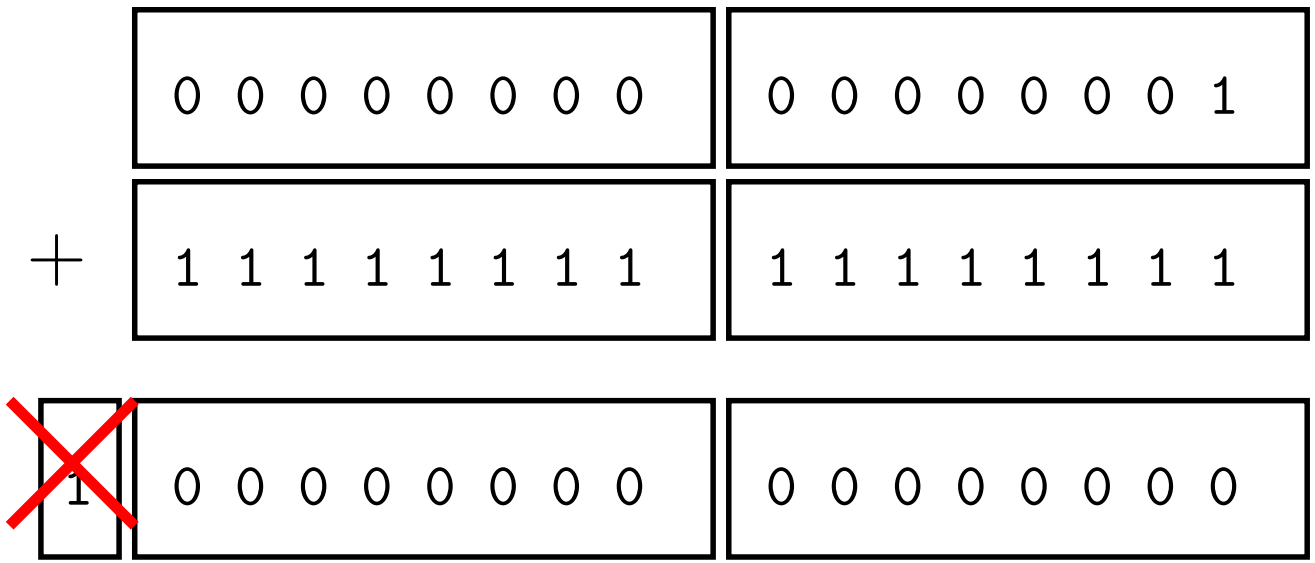
c) ni l'un, ni l'autre

En utilisant la règle de signe supposée, on a (au moins) deux écritures possibles pour zéro !

1 0 0 0 0 0 0 0	0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

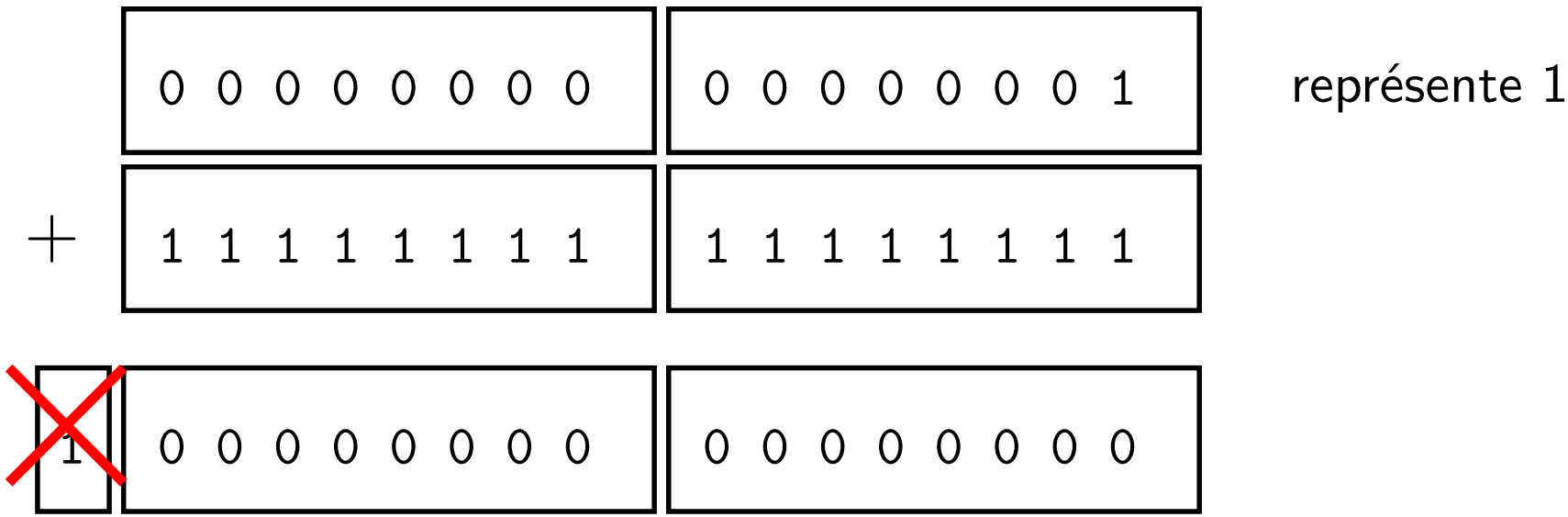
Signe par le complément

L'ordinateur utilise la règle d'addition (lacunaire)



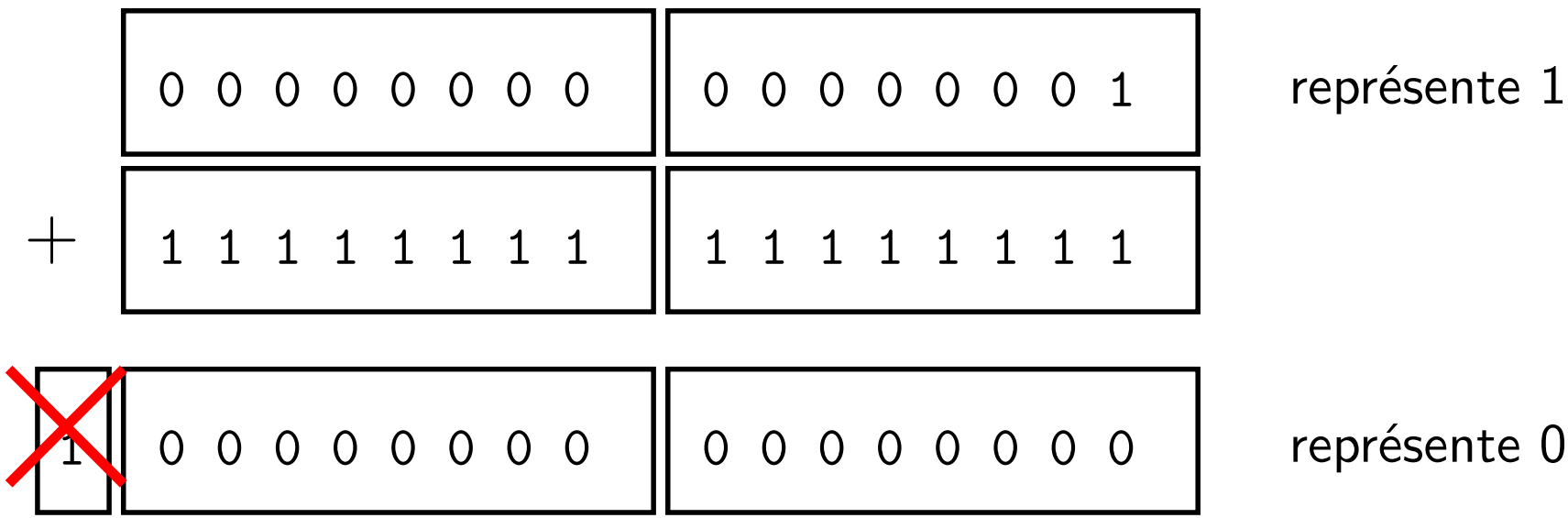
Signe par le complément

L'ordinateur utilise la règle d'addition (lacunaire)



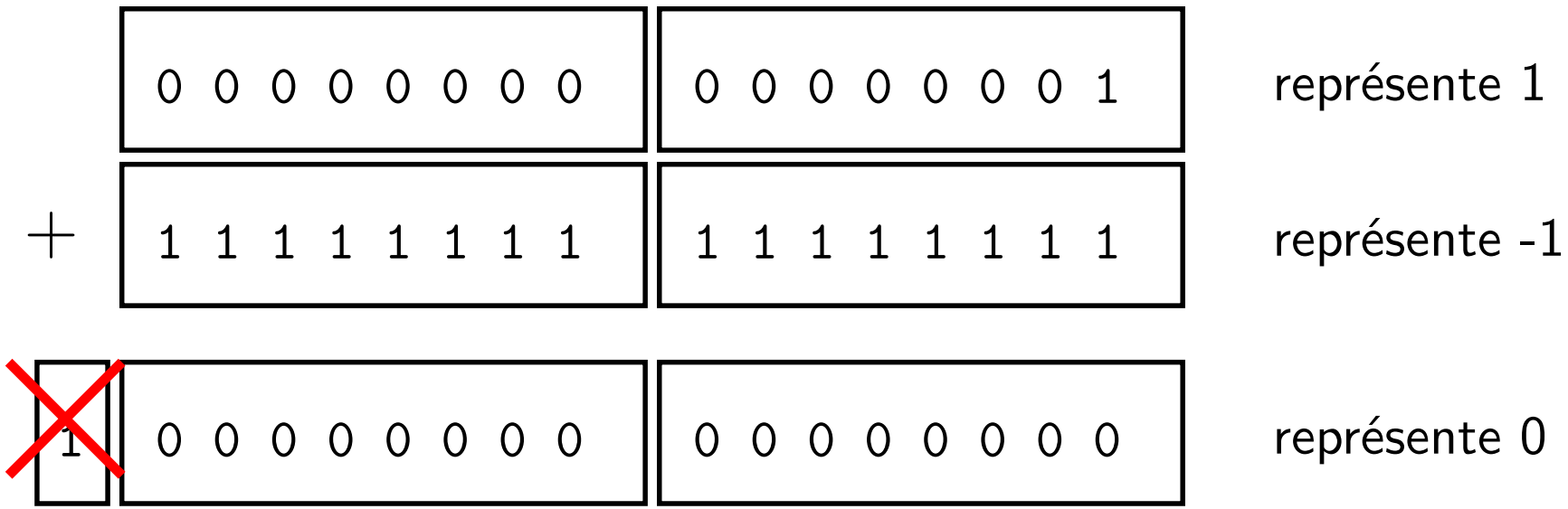
Signe par le complément

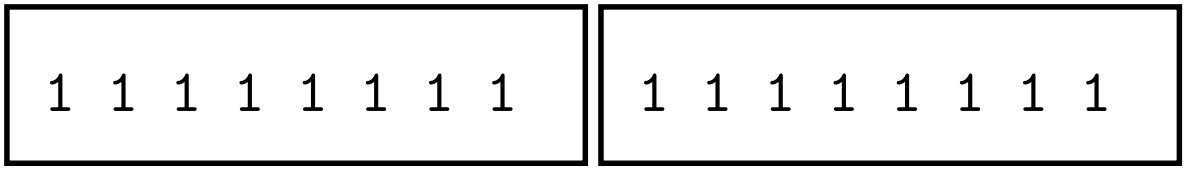
L'ordinateur utilise la règle d'addition (lacunaire)



Signe par le complément

L'ordinateur utilise la règle d'addition (lacunaire)





Que représente en décimal cette séquence de 2 octets ?

a) -1

b) $2^{16} - 1 = 65535$

c) ni a), ni b)

Résultat à l'exécution des lignes suivantes

```
1  short s4 = -1;
2  unsigned short s5 = s4;
3
4  printf("%d ", s4);
5  printf("%d \n", s5);
```

a) -1 65535

b) -1 -1

c) 65535 65535

d) un autre résultat

Résultat à l'exécution des lignes suivantes

```
1 char ch1 = 'A';
2 short s1 = ch1;
3 printf("%d %d\n", ch1, s1);
```

a) A A

b) A 65

c) un autre résultat

Résultat à l'exécution des lignes suivantes

```
1 short s6 = 5;
2 char ch6 = s6;
3 printf("%d %d\n", ch6, s6);
```

a) 5 5

b) 5 36560

c) un autre résultat

Résultat à l'exécution des lignes suivantes

```
1  short s7 = 129;
2  char ch7 = s7;
3  printf("%d %d\n", ch7, s7);
```

a) 129 129

b) -127 129

c) -1 129

d) un autre résultat

Résultat à l'exécution des lignes suivantes

```
1  short s8 = 258;
2  char ch8 = s8;
3  printf("%d %d\n", ch8, s8);
```

a) 2 258

b) -123 258

c) -2 258

d) un autre résultat

Résultat à l'exécution des lignes suivantes

```
1  short s9 = -1;
2  int i = s9;
3  printf("%d %d\n", s9, i);
```

a) 65535 -1

b) -65535 -1

c) -1 -1

d) un autre résultat

Représentation des flottants

Nombres flottants : représentation « approchée » des nombres réels.

$$x \in \mathbb{R}, \qquad x \simeq x^{\text{float}}.$$

x^{float} est un nombre que l'on peut représenter exactement grâce à un nombre fini de bits/octets.

```
float f;
double d;
```

On considère par la suite que

- les `float` sont stockés sur 4 octets
- les `double` sont stockés sur 8 octets

(cela peut dépendre de la machine et des compilateurs)

Représentation des flottants

Nombres flottants : représentation « approchée » des nombres réels.

$$x \in \mathbb{R}, \qquad x \simeq x^{\text{float}}.$$

x^{float} est un nombre que l'on peut représenter exactement grâce à un nombre fini de bits/octets.

```
float f;
double d;
```

On considère par la suite que

- les `float` sont stockés sur 4 octets
- les `double` sont stockés sur 8 octets

(cela peut dépendre de la machine et des compilateurs)

Représentation des flottants

Nombres flottants : représentation « approchée » des nombres réels.

$$x \in \mathbb{R}, \qquad x \simeq x^{\text{float}}.$$

x^{float} est un nombre que l'on peut représenter exactement grâce à un nombre fini de bits/octets.

```
float f;
double d;
```

On considère par la suite que

- les `float` sont stockés sur 4 octets
- les `double` sont stockés sur 8 octets

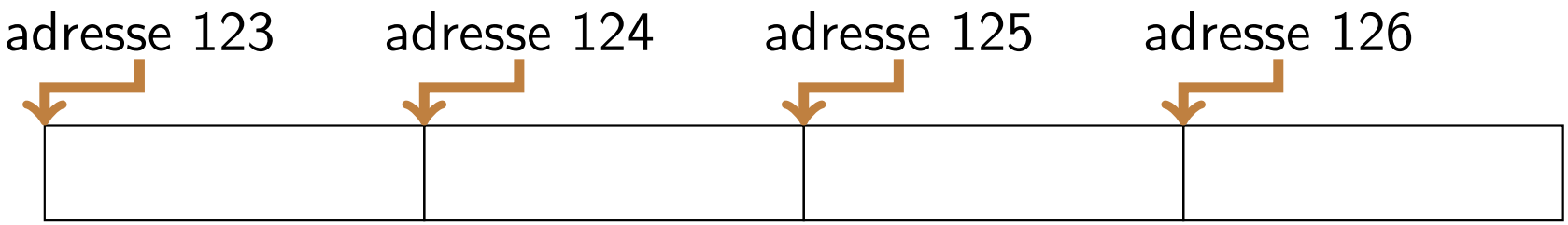
(cela peut dépendre de la machine et des compilateurs)

Comment cela se passe-t'il ?

$$x \simeq x^{\text{float}} = (-1)^{s}\left\{1 + \frac{a_1}{2^1} + \cdots + \frac{a_{23}}{2^{23}}\right\} \times 2^{(b_0 \times 2^0 + b_1 \times 2^2 + \ldots + b_7 \times 2^7) - 127}.$$

s, a_1, $\ldots$, a_{23}, b_0, $\ldots$, b_7 sont tous des éléments de $\{0, 1\}$ (32 bits)

```
float f=-10.3;
```



a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}
a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
b_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
s	b_7	b_6	b_5	b_4	b_3	b_2	b_1

bits de l'octet d'adresse 123

bits de l'octet d'adresse 124

bits de l'octet d'adresse 125

bits de l'octet d'adresse 126

$$-10.3 =$$

$$2^3 < 10.3 < 2^4$$

$$-10.3 =$$

$$2^3 < 10.3 < 2^4$$

$$-10.3 = -1.2875 \times 2^3$$

$$2^3 < 10.3 < 2^4$$

$$\begin{aligned} -10.3 &= -1.2875 \times 2^3 \\ &= (-1)^1 \times \left(1 + \sum_{k=1}^{+\infty} \frac{a_k}{2^k}\right) \times 2^{130-127} \end{aligned}$$

$$2^3 < 10.3 < 2^4$$

$$\begin{aligned} -10.3 &= -1.2875 \times 2^3 \\ &= (-1)^1 \times \left(1 + \sum_{k=1}^{+\infty} \frac{a_k}{2^k}\right) \times 2^{130-127} \end{aligned}$$

$$\begin{cases} 130 = 2^2 + 2^7 \\ 0.2875 = \sum_{k=1}^{+\infty} \frac{a_k}{2^k} \end{cases}$$

$$
\begin{cases}
0.2875 = \displaystyle\sum_{k=1}^{+\infty} \frac{a_k}{2^k} \\
\quad = \dfrac{0}{2^1} + \dfrac{1}{2^2} + \dfrac{0}{2^3} + \dfrac{0}{2^4} + \dfrac{1}{2^5} + \dfrac{0}{2^6} + \dfrac{0}{2^7} + \dfrac{1}{2^8} \\
\quad + \dfrac{1}{2^9} + \dfrac{0}{2^{10}} + \dfrac{0}{2^{11}} + \dfrac{1}{2^{12}} + \dfrac{1}{2^{13}} + \dfrac{0}{2^{14}} + \dfrac{0}{2^{15}} + \dfrac{1}{2^{16}} \\
\quad + \dfrac{1}{2^{17}} + \dfrac{0}{2^{18}} + \dfrac{0}{2^{19}} + \dfrac{1}{2^{20}} + \dfrac{1}{2^{21}} + \dfrac{0}{2^{22}} + \dfrac{0}{2^{23}} + \cdots
\end{cases}
$$

$$
\begin{cases}
0.2875 = \displaystyle\sum_{k=1}^{+\infty} \frac{a_k}{2^k} \\
\quad = \dfrac{0}{2^1} + \dfrac{1}{2^2} + \dfrac{0}{2^3} + \dfrac{0}{2^4} + \dfrac{1}{2^5} + \dfrac{0}{2^6} + \dfrac{0}{2^7} + \dfrac{1}{2^8} \\
\quad + \dfrac{1}{2^9} + \dfrac{0}{2^{10}} + \dfrac{0}{2^{11}} + \dfrac{1}{2^{12}} + \dfrac{1}{2^{13}} + \dfrac{0}{2^{14}} + \dfrac{0}{2^{15}} + \dfrac{1}{2^{16}} \\
\quad + \dfrac{1}{2^{17}} + \dfrac{0}{2^{18}} + \dfrac{0}{2^{19}} + \dfrac{1}{2^{20}} + \dfrac{1}{2^{21}} + \dfrac{0}{2^{22}} + \dfrac{0}{2^{23}} + \cdots \\
\quad \simeq \dfrac{0}{2^1} + \dfrac{1}{2^2} + \dfrac{0}{2^3} + \dfrac{0}{2^4} + \dfrac{1}{2^5} + \dfrac{0}{2^6} + \dfrac{0}{2^7} + \dfrac{1}{2^8} \\
\quad + \dfrac{1}{2^9} + \dfrac{0}{2^{10}} + \dfrac{0}{2^{11}} + \dfrac{1}{2^{12}} + \dfrac{1}{2^{13}} + \dfrac{0}{2^{14}} + \dfrac{0}{2^{15}} + \dfrac{1}{2^{16}} \\
\quad + \dfrac{1}{2^{17}} + \dfrac{0}{2^{18}} + \dfrac{0}{2^{19}} + \dfrac{1}{2^{20}} + \dfrac{1}{2^{21}} + \dfrac{0}{2^{22}} + \dfrac{1}{2^{23}}
\end{cases}
$$

$$-10.3 \simeq (-1)^s \times \left(1 + \sum_{k=1}^{23} \frac{a_k}{2^k}\right) \times 2^{(b_0 + b_1 2 + \ldots + b_7 2^7) - 127}$$

$$s = 1$$

$$b_1 = 1, \quad b_7 = 1, \quad \text{sinon } b_k = 0.$$

$$a_2 = 1, \quad a_5 = 1, \quad a_8 = 1, \quad a_9 = 1, \quad a_{12} = 1, \quad a_{13} = 1$$
$$a_{16} = 1, \quad a_{17} = 1, \quad a_{20} = 1, \quad a_{21} = 1, \quad a_{23} = 1,$$

$$\text{sinon } a_k = 0$$

On obtient la séquence d'octets suivante

1 1 0 0 1 1 0 1	1 1 0 0 1 1 0 0	0 0 1 0 0 1 0 0	1 1 0 0 0 0 0 1

- Le premier octet peut-être lu comme $2^0 + 2^2 + 2^3 + 2^6 + 2^7 = 205$
- Le deuxième octet peut-être lu comme $2^2 + 2^3 + 2^6 + 2^7 = 204$
- Le troisième octet peut-être lu comme $2^2 + 2^5 = 36$
- Le quatrième octet peut-être lu comme $2^0 + 2^6 + 2^7 = 193$

Les règles d'opérations arithmétiques standard (+,-, *, etc.) sont totalement différentes (en terme de manipulation de bits) pour les entiers et les flottants !