Introduction à l'informatique

Les opérations

Jean-Christophe Dubacq

IUT de Villetaneuse

S1 2016

Plan

Plan

Exercices

De la fonction à l'algorithme

La numération grecque (simple) est proche de la numération romaine que vous connaissez : on note les nombres comme suit :

1	5	10	50	100	500	1 000	5 000	10 000	50 000
I	Γ	Δ	Γ_Δ	H	Γ_H	X	Γ_X	M	Γ_M

C'est à la différence près que l'on a pas de règle soustractive : le nombre 4 s'écrit IIII, pas $\mathrm{I}\Gamma$. La position des chiffres n'a théoriquement aucune importance, mais on les classait dans l'ordre décroissant de valeur.

Q1 Ce système est-il un système de numération positionnelle ?

Q2 Écrivez votre âge et votre date de naissance en numération grecque.

Q3 Écrivez un algorithme d'addition des nombres représentés en numération grecque. Est-ce que cet algorithme est le même qu'en décimal ?

Q4 Faites l'addition de votre âge et de votre année de naissance avec votre algorithme (vous devriez obtenir $\mathrm{XX}\Delta\mathrm{IIII}$ ou $\mathrm{XX}\Delta\mathrm{III}$). De quelle représentations partez-vous ?

Q5 Faites la même chose en décimal. De quelles représentations partez-vous ? Est-ce que l'algorithme est le même ? Est-ce que la fonction calculée est la même ?

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$, *où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

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

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 1

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

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 2

$$
\begin{array}{cccccc}
 & & & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 \\
+ & & 1 & 1 & 1 & 1 \\
\hline
 & & & & \boxed{10} & 1
\end{array}
$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 2

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

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 4

$$
\begin{array}{cccccc}
 & & & 1 & & \\
 & & & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 \\
+ & & 1 & 1 & 1 & 1 \\
\hline
 & & & \boxed{100} & 0 & 1
\end{array}
$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base $\mathtt{B}$ *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathtt{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* $\mathtt{B}$ *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 4

$$\begin{array}{cccccc}
 & & 10 & 1 & & \\
 & & & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 \\
+ & & 1 & 1 & 1 & 1 \\
\hline
 & & & 0 & 0 & 1
\end{array}$$

On peut aussi marquer la retenue $\mathtt{10}$ avec $\mathtt{0}$ dans la colonne suivante et $\mathtt{1}$ dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

$$
\begin{array}{ccccccc}
 & & & 10 & 1 & & \text{poids 8} \\
 & & & & 1 & 0 & 0 \\
+ & 1 & & 0 & 1 & 1 & 0 \\
+ & & & 1 & 1 & 1 & 1 \\
\hline
 & & & \boxed{11} & 0 & 0 & 1
\end{array}
$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

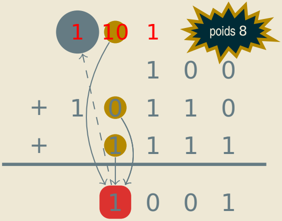
Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)



On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + Br$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

$$
\begin{array}{cccccc}
 & 1 & 10 & 1 & & \\
 & & & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 \\
+ & & 1 & 1 & 1 & 1 \\
\hline
 & \boxed{10} & 1 & 0 & 0 & 1
\end{array}
$$

poids 16

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

1	1	10	1	poids 16		
			1	0	0	
+	1	0	1	1	0	
+		1	1	1	1	
	0	1	0	0	1	

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

poids 32

$$\begin{array}{cccccc} 1 & 1 & 10 & 1 & & \\ & & & 1 & 0 & 0 \\ + & 1 & 0 & 1 & 1 & 0 \\ + & & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \end{array}$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Addition dans les systèmes positionnels

Méthode (Addition)

L'addition en base B *se fait de la droite vers la gauche, colonne par colonne, en utilisant le fait que la somme de chiffres s'écrit sous la forme* $s + \mathrm{B}r$*, où* s *est la somme partielle (un chiffre unique) et* r *est la retenue. Le poids de* r *est* B *fois plus important, et* r *est donc remise dans la colonne d'à côté.*

Exemple (Addition en base 2)

$$
\begin{array}{cccccc}
\color{red}{1} & \color{red}{1} & \color{red}{10} & \color{red}{1} & & \\
 & & & 1 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 \\
+ & & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 1 & 0 & 0 & 1
\end{array}
$$

On peut aussi marquer la retenue 10 avec 0 dans la colonne suivante et 1 dans la colonne d'ordre encore supérieur.

Truc : pour additionner une colonne, on peut bien sûr le faire en décimal, à condition de repasser au binaire pour le reste et les retenues.

Multiplication

Méthode (Multiplication)

À l'instar de l'addition, la multiplication se fait comme pour les nombres décimaux. Les tables sont justes différentes (il faut les écrire dans la bonne base !).
En binaire, c'est très facile : on multiplie par 0 ou par 1, donc on se contente d'additionner des copies du multiplicande décalées là où le multiplicateur a des 1.
***Très important :** décaler à gauche de n colonnes un nombre, c'est le multiplier par B^n. Le décaler à droite, c'est le diviser par B^n.*

$$
\begin{array}{cccccc}
 & & & 1 & 1 & 0 \\
\times & & & 1 & 0 & 1 \\
\hline
 & & & 1 & 1 & 0 \\
(+ & & & & 0 &) \\
+ & 1 & 1 & 0 & & \\
\hline
 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

Négatifs et réels

Par analogie avec les techniques maîtrisées en base 10, il est possible de faire également :

- Des soustractions : lorsque le chiffre duquel on soustrait n'est pas suffisant, on ajoute 1 au chiffre à soustraire dans la colonne d'ordre immédiatement supérieur, et en compensation, on ajoute la base dans la colonne courante.
- Des divisions : en fait, on fait plein de multiplications enchaînées avec des soustractions.
- Additionner un négatif, c'est soustraire un positif.
- Des opérations en virgule fixe : les règles de placement de la virgule sont les mêmes qu'en base 10.
- Des décalages qui sont des multiplications ou divisions par une puissance de la base.
- Pour la virgule flottante, les multiplications sont simples ; les additions nécessitent de recoder en virgule fixe.

Exercices

Calcul en binaire et hexadécimal

Q6 Faites les additions en binaire : `0b1101 0101`+`0b1110 0101` ; `0b1,1`+`0b110`+`0b100,1`+`0b111,1`+`0b1010,1`+`0b100,1`.

Q7 Faites les opérations suivantes en hexadécimal : `0x122` + `0x233` ; `0x87` + `0x54` ; `0x18` + `0x9` ; `0xED` + `0xED` ; `0x100` − `0x3`.

Q8 Faites la multiplication suivante : `17` × `129` à la fois en décimal et en binaire.

Q9 Faites la multiplication suivante en binaire : `110110` × `1101`.

Plan

Les algorithmes d'addition

- L'addition de codage se fait par un algorithme (représentation/information). Si l'algorithme fait ce qu'on attend de lui (identique à la fonction), il est *correct*.
- Tous les nombres ne sont pas représentables. Si le résultat de l'addition (fonction) donne un nombre non représentable, l'algorithme ne peut pas être correct.
- L'algorithme d'addition pour NAT et C2 est très similaire à la méthode usuelle (décrite avant) ; on coupe juste le résultat à la taille du codage.

Théorème

En C2 et NAT, si le résultat est représentable, l'addition (classique) est correcte.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xRightarrow{\text{code}} 1000\,0011_{NAT} + 0101\,0101_{NAT}$$

$$\overset{\text{algo}}{=}$$

$$216 \xLeftarrow{\text{decode}} 1101\,1000_{NAT}$$

$$187 + 101 \xRightarrow{\text{code}} 1011\,1011_{NAT} + 0110\,0101_{NAT}$$

$$\overset{\text{algo}}{=}$$

$$32 (\neq 288) \xLeftarrow{\text{decode}} 0010\,0000_{NAT}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{NAT} + 0101\,0101_{NAT}$$

$$\overset{\text{algo}}{=}$$

Correct !

$$216 \xleftarrow{\text{decode}} 1101\,1000_{NAT}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{NAT} + 0110\,0101_{NAT}$$

$$\overset{\text{algo}}{=}$$

$$32(\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{NAT}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \overset{\text{code}}{\Longrightarrow} 1000\,0011_{\mathrm{NAT}} + 0101\,0101_{\mathrm{NAT}}$$

Correct !

$$\overset{\text{algo}}{=}$$

$$216 \overset{\text{decode}}{\Longleftarrow} 1101\,1000_{\mathrm{NAT}}$$

$$187 + 101 \overset{\text{code}}{\Longrightarrow} 1011\,1011_{\mathrm{NAT}} + 0110\,0101_{\mathrm{NAT}}$$

$$\overset{\text{algo}}{=}$$

$$32 (\neq 288) \overset{\text{decode}}{\Longleftarrow} 0010\,0000_{\mathrm{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Addition correcte ou pas ?

Exemple (Addition en NAT)

$$131 + 85 \xrightarrow{\text{code}} 1000\,0011_{\text{NAT}} + 0101\,0101_{\text{NAT}}$$

Correct ! $\quad \overset{\text{algo}}{=}$

$$216 \xleftarrow{\text{decode}} 1101\,1000_{\text{NAT}}$$

$$187 + 101 \xrightarrow{\text{code}} 1011\,1011_{\text{NAT}} + 0110\,0101_{\text{NAT}}$$

Incorrect ! $\quad \overset{\text{algo}}{=}$

$$32 (\neq 288) \xleftarrow{\text{decode}} 0010\,0000_{\text{NAT}}$$

- En C1 et en VA+S, il suffit de mettre un négatif pour opération incorrecte
- En plus, problème du double zéro pour C1 et VA+S.
- C2 : comme si le bit de poids fort était de poids -2^{n-1} au lieu de 2^{n-1}, arithmétique modulo 2^n.

Notes sur la multiplication et l'hexadécimal

- La multiplication est extrêmement simple en binaire. Elle se comporte comme une série d'additions.
- La multiplication est rarement correcte si on a autant de chiffres en entrée qu'en sortie. La plupart des ordinateurs multiplient en mettant le résultat dans deux mots de code distincts (partie haute et partie basse).

Exercices

Limites de la multiplication

Expliquez pourquoi le résultat d'une multiplication de deux nombres représentés dans l'un des 4 codes classiques est toujours représentable à condition de doubler la taille du code.

Addition en C2

Q10 Faites les opérations suivantes en transformant les nombres au préalable en codage C2 sur 8 bits (résultat aussi en C2 sur 8 bits) :

- 45+17
- 45-17 (soit 45+(-17))
- -17-17
- 17-45
- 221+45

Dites aussi si le résultat obtenu est correct et s'il est représentable.

Plan

La logique booléenne

- Il est possible d'interpréter les deux valeurs binaires comme représentant respectivement *vrai* (1) ou *faux* (0).
- On peut définir des opérations correspondantes à la conjonction (et), la disjonction (ou) et la négation (opposé de).

Ce ne sont pas des opérations arithmétiques classiques.

$c = ab$ ou $c = a \wedge b$ pour le *et* (en C : $a = b\&c$)

$c = a + b$ ou $c = a \vee b$ pour le *ou* (en C : $a = b|c$)

$b = \overline{a}$ ou $b = \neg a$ pour le *non* (en C : $a =\sim c$)

Exemple (chemin sous UNIX)

Soit $\mathcal{A}(p)$ la propriété « p est un chemin existant » et $\mathcal{B}(p)$ la propriété « p est un chemin qui désigne un répertoire ». Si on suppose qu'il n'y a que deux type d'objets (répertoires et fichiers), la propriété $\mathcal{C}(p)$ « p est un chemin qui désigne un fichier » s'exprime par $\mathcal{A}(p)\overline{\mathcal{B}(p)}$.

Les opérateurs booléens

AND et OR

AND

L'opérateur binaire AND, noté $a \times b$, renvoie 1 si et seulement si ses deux arguments sont égaux à 1.
L'opérateur général AND renvoie 1 si et seulement si tous ses arguments sont égaux à 1.
Ils sont équivalents à la fonction *minimum*.

OR

L'opérateur binaire OR, noté $a + b$, renvoie 0 si et seulement si ses deux arguments sont égaux à 0.
L'opérateur général OR renvoie 0 si et seulement si tous ses arguments sont égaux à 0.
Ils sont équivalents à la fonction *maximum*.

Les opérateurs booléens

XOR et NOT

XOR

L'opérateur binaire XOR, noté $a \oplus b$, renvoie 1 si et seulement si ses deux arguments sont différents.
Ils sont équivalents à la fonction *différent*.

NOT

L'opérateur unaire NOT, noté $\overline{a}$, renvoie 0 si et seulement si son argument est égal à 1.
Il est équivalent à la fonction *complémentation*.
Il y a aussi les opérateurs généraux NOR et NAND qui sont en fait NOT(OR(...)) et NOT(AND(...)). Ils sont rarement implémentés dans les langages de programmation.

Exercices

Tables de vérité

Q11 Faites une table qui montre toutes les paires d'arguments possibles pour les opérateurs AND, OR, XOR et qui montre le résultat à côté.

A	B	$A \times B$
0	0	
0	1	
1	0	
1	1	

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

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

Les champs de bits

- La notion de variable booléenne stockant une valeur vraie ou faux est très souvent intégrée directement dans les langages

Ce n'est pas le cas dans le langage C

- On appelle parfois ces variables des *flags* (drapeaux).
- Quand on réunit plusieurs de ces variables dans une même entité, on appelle le résultat un champ de bits (anglais *bit field*).
- Ces champs de bits peuvent être stockés dans une seule variable (selon leur nombre). On les considère comme un entier codé en NAT ou C2.
- On définit des opérations sur les champs de bits : le *et bit-à-bit*, le *ou bit-à-bit*, le *not bit-à-bit* et le *xor bit-à-bit*.

Il s'agit de faire sur les bits de même position dans deux champs de bits (un pour la négation) l'opération booléenne correspondante.

Exercices

Opérations booléennes

Q12 Que vaut 0b10000110 AND 0b11101001 ?

Q13 Que vaut 0b10000110 OR 0b11101001 ?

Q14 Que vaut 0b10000110 XOR 0b11101001 ?

Q15 Que se passe-t-il si on calcule (a est une variable booléenne) : $a + 0$? $a + 1$? $a \times 0$? $a \times 1$? $a + a$? $a + a + a + a + a + a$?

Q16 Démontrez que $a + ab = a$;

Q17 Démontrez que $a + bc = (a + b)(a + c)$;

Q18 Démontrez que $a + \overline{a}b = a + b$;

Masquage

Lorsqu'un champ de bits est représenté par un entier, on peut accéder à un bit particulier en procédant à un ET :

$$b_x = (\mathtt{B}\&(1 << x)) >> x$$

On obtient 1 si le bit numéro x est à 1, 0 sinon.
On peut aussi mettre à 1 le bit numéro x :

$$\mathtt{B} = \mathtt{B}|(1 << x)$$

ou à zéro :

$$\mathtt{B} = \mathtt{B}\&(\sim (1 << x))$$

On peut aussi inverser le bit numéro x :

$$\mathtt{B} = \mathtt{B} \oplus (1 << x)$$

On peut tester si par exemple le bit 1 ou 3 sont à 1 : `if (a&0b1010 != 0) ...`
L'ensemble de ces techniques pour manipuler un champ de bits sous la forme d'un entier est appelé *masquage*.
Souvent, les valeurs $(1 << x)$ sont nommées pour qu'on puisse simplement utiliser leur nom au lieu de se souvenir de leur position.

Exercices

Analyse d'un masquage

Dans un champ de bits qui contient $a = 0b11001001$, on veut faire les choses suivantes :

Q19 On veut vérifier si le bit 0 est actif ou non. Décomposez l'opération.

Q20 On veut changer le bit 1 en 1 et le bit 3 en 0. Décomposez les opérations qui permettent de le faire.

Q21 Changez le bit 5, en expliquant les valeurs intermédiaires.

Analyse de touches

Dans un système, la fonction `keyEvent()` renvoie une valeur entière sur 16 bits (dont 5 ignorés) :

- Les 8 premiers bits correspondent au numéro de la touche sur le clavier (pour les touches ordinaires)
- Le 9e bit correspond à la touche SHIFT (1 : pressée, 0 : pas pressée)
- Le 10e bit correspond à la touche CONTROL (1 : pressée, 0 : pas pressée)
- Le 11e bit correspond au fait d'appuyer sur une touche (1) ou de l'avoir juste relachée (0)

Q22 Écrivez un programme qui appelle cette fonction (`a=keyEvent()`) puis qui en fonction de `a` affiche un texte du genre : « Vous venez de lâcher la touche 27 en ayant SHIFT appuyé et CONTROL lâché »