Compléments sur les littéraux entiers

Par défaut les littéraux entiers sont écrits en décimal.

Ils peuvent aussi être écrits en hexadécimal ou en octal s'ils commencent par `0x` ou `0`

25	
0x19	25 en hexadécimal (16 + 9)
031	25 en octal (3 * 8 + 1)

Rappels sur l'algèbre de Boole (1)

L'opérateur non

x	$\overline{x}$
0	1
1	0

Rappels sur l'algèbre de Boole (2)

L'opérateur et

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

$$x \cdot 0 = 0$$
$$x \cdot 1 = x$$

Rappels sur l'algèbre de Boole (3)

L'opérateur ou

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

$$x + 0 = x$$
$$x + 1 = 1$$

Rappels sur l'algèbre de Boole (4)

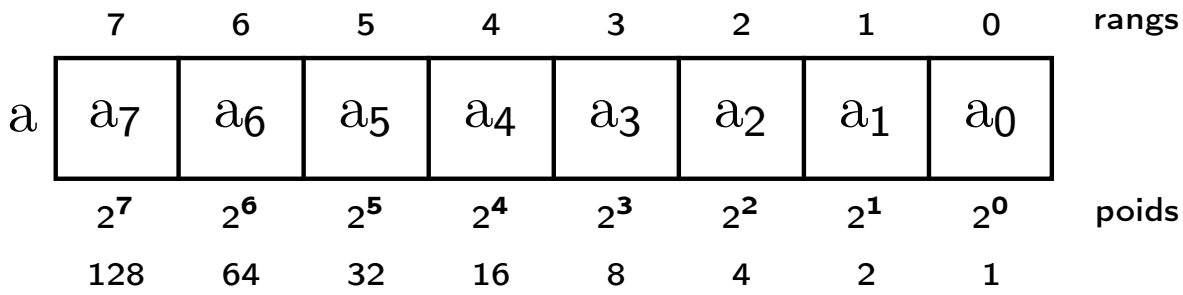
L'opérateur ou exclusif

x	y	x $\oplus$ y
0	0	0
0	1	1
1	0	1
1	1	0

$$x \oplus 0 = x$$
$$x \oplus 1 = \overline{x}$$

Le système binaire

Soit **a** un octet :



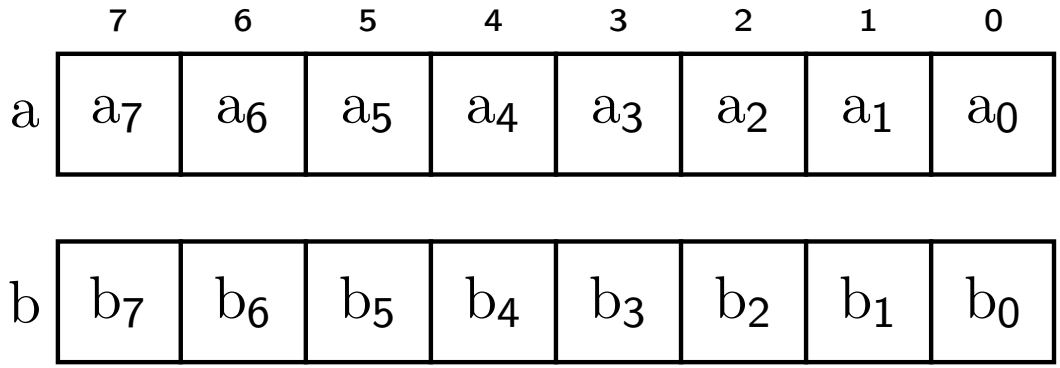
Le bit de rang i a pour poids 2^i

Les opérateurs bits à bits

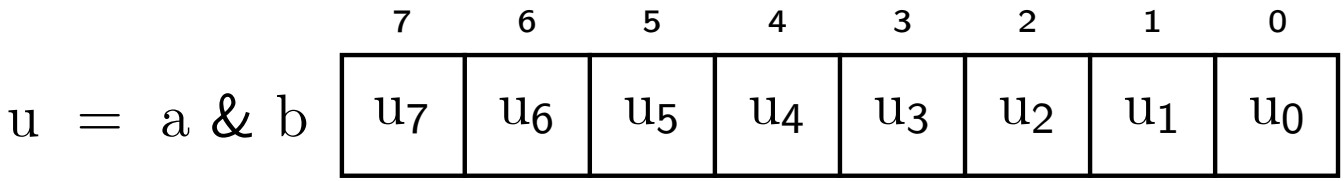
&	et bits à bits
|	ou bits à bits
^	ou exclusif bits à bits
~	complément

Le et bits à bits &

Soient **a** et **b** deux octets (ou plus généralement des mots de n bits)



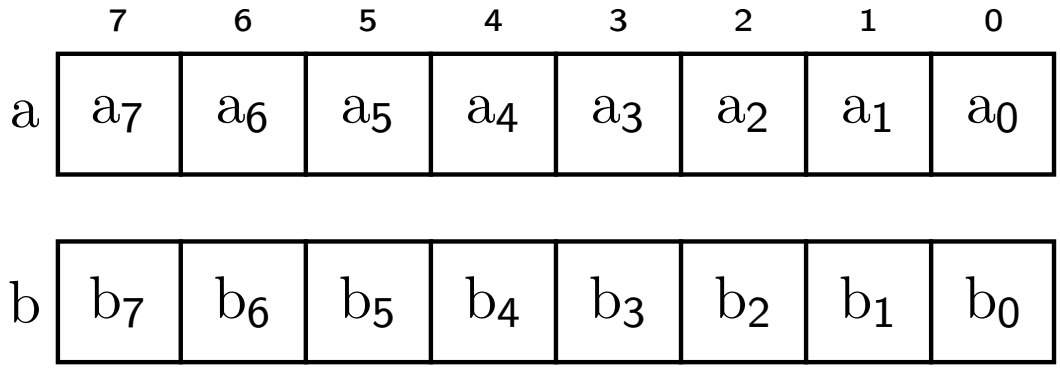
Le **et** bits à bits entre les octets **a** et **b** est :



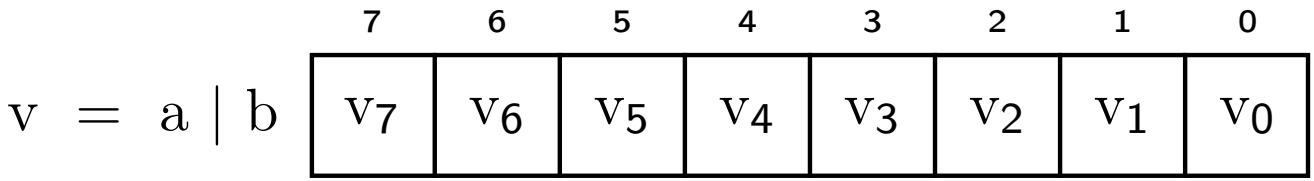
avec $u_i = a_i \cdot b_i$

Le ou bits à bits |

Soient **a** et **b** deux octets (ou plus généralement des mots de n bits)



Le **ou** bits à bits entre les octets **a** et **b** est :



avec $v_i = a_i + b_i$

Le ou exclusif bits à bits ^

Soient **a** et **b** deux octets (ou plus généralement des mots de n bits)

	7	6	5	4	3	2	1	0
a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
b	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

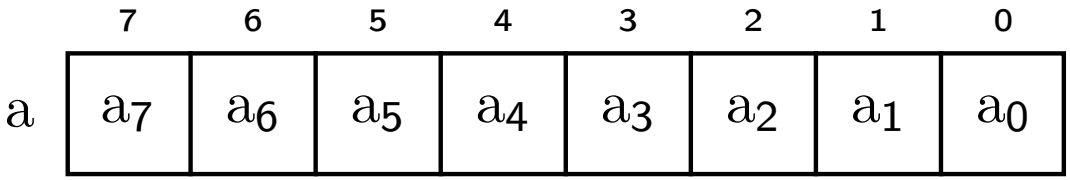
Le **ou** exclusif bits à bits entre les octets **a** et **b** est :

	7	6	5	4	3	2	1	0
$w = a \hat{} b$	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0

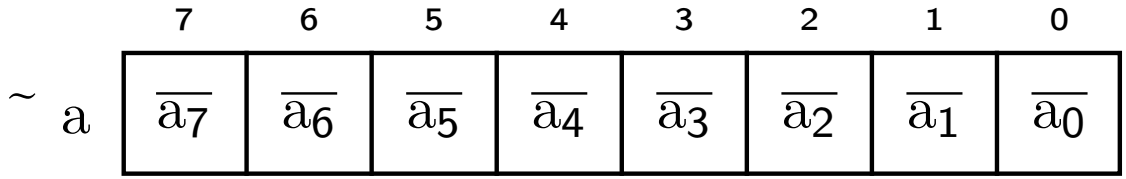
avec $w_i = a_i \oplus b_i$

Le complément ~

Le complément de l'octet **a**

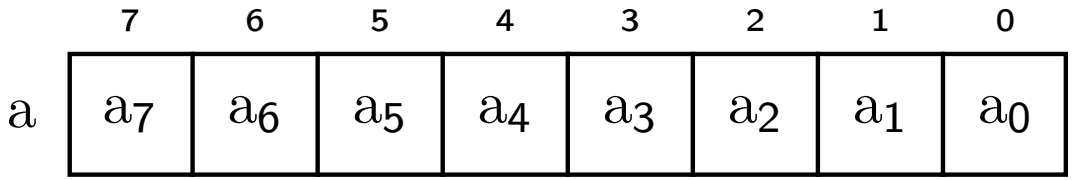


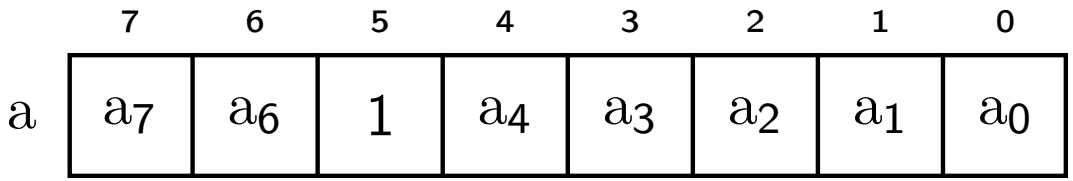
est l'octet



Exercice 1

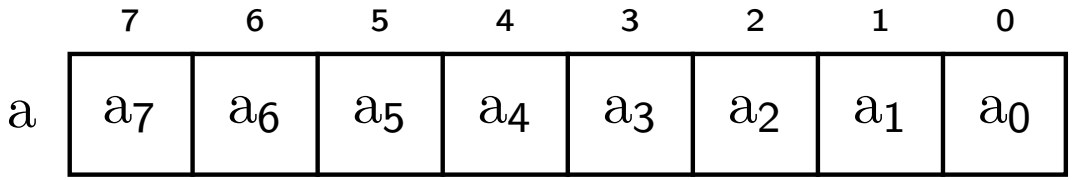
Donner l'instruction qui permet de forcer à 1 le bit de rang 5 de l'octet **a**

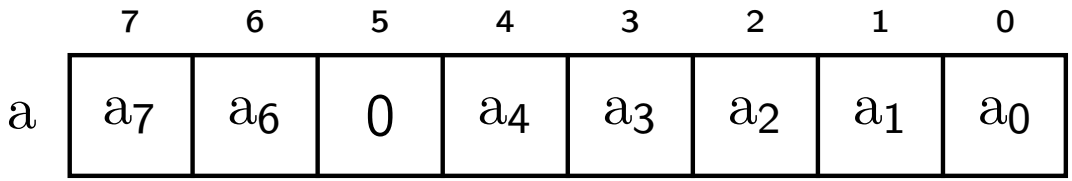




Exercice 2

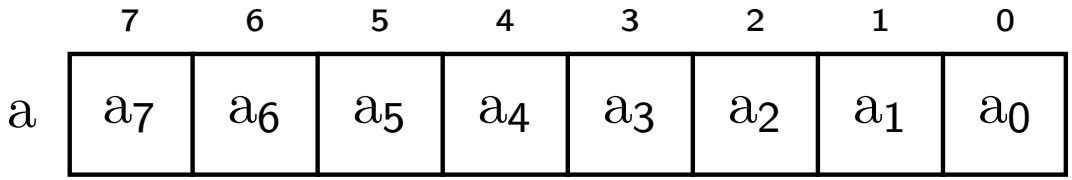
Donner l'instruction qui permet de forcer à 0 le bit de rang 5 de l'octet **a**

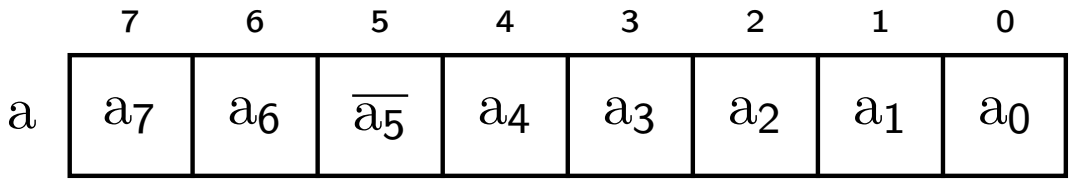




Exercice 3

Donner l'instruction qui permet de complémenter le bit de rang 5 de l'octet **a**



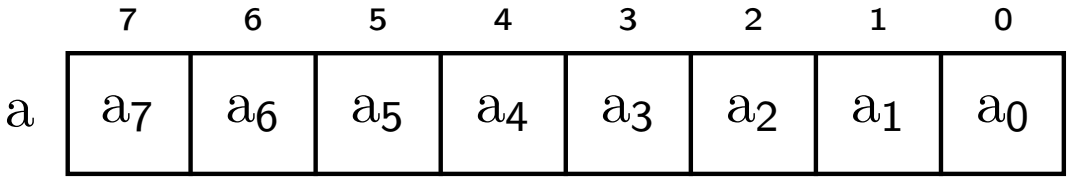


Les opérateurs de décalage

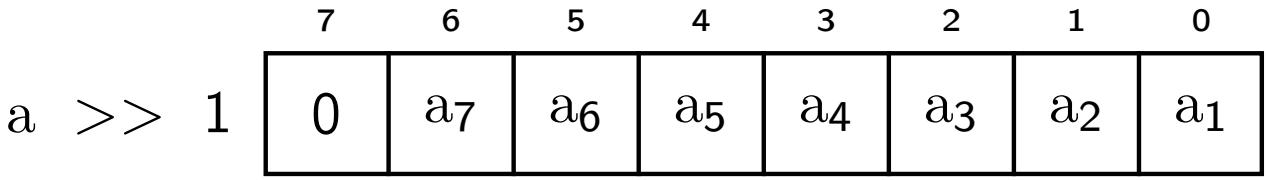
>>	décalage à droite
<<	décalage à gauche

Décalage à droite (1)

Le décalage à droite d'une position de l'octet **a**

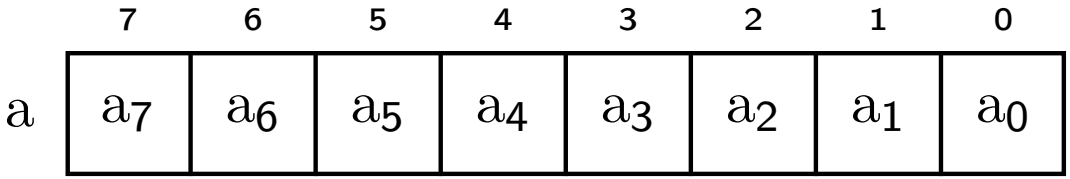


est :

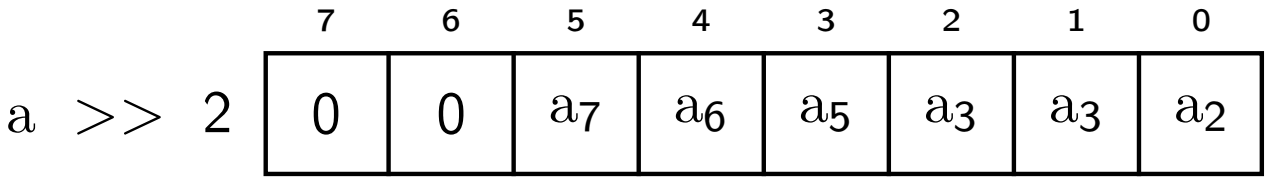


Décalage à droite (2)

Le décalage à droite de deux positions de l'octet **a**

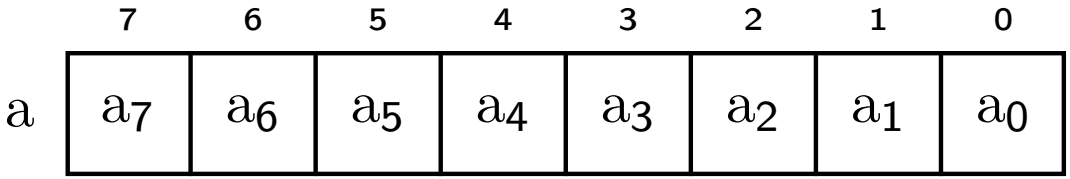


est :

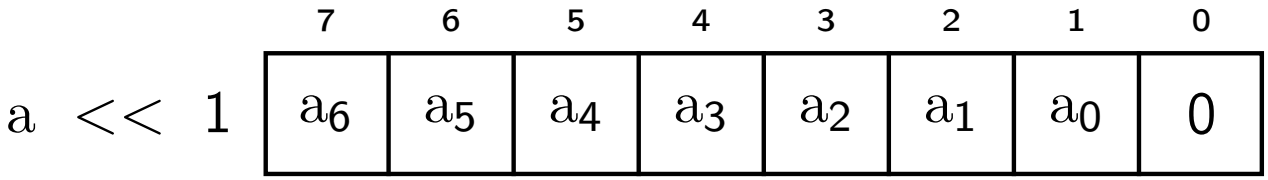


Décalage à gauche (1)

Le décalage à gauche d'une position de l'octet **a**

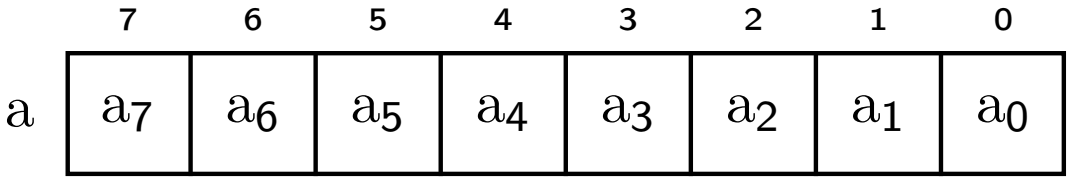


est :

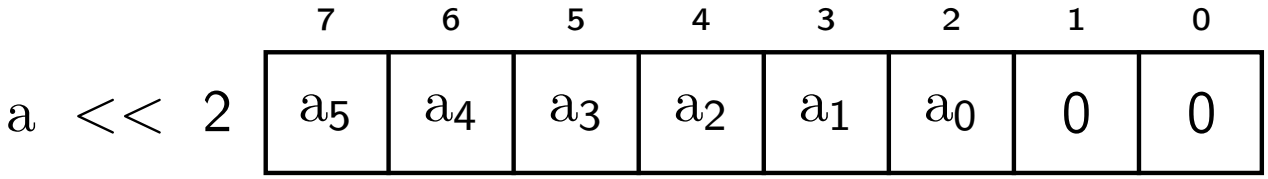


Décalage à gauche (2)

Le décalage à gauche de deux positions de l'octet **a**



est :



La fonction quartetPoidsFaible

Ecrire la fonction **quartetPoidsFaible** qui retourne le quartet de poids faible de l'octet passé en paramètre.

```
quartetPoidsFaible ( 25 ) retourne  9
                          ( 25 = 0x19 )
```

```
public static int quartetPoidsFaible( int n ){

}
```

La fonction quartetPoidsFort

Ecrire la fonction **quartetPoidsFort** qui retourne le quartet de poids fort de l'octet passé en paramètre.

```
quartetPoidsFort ( 25 ) retourne  1
                        ( 25 = 0x19 )
```

```
public static int quartetPoidsFort( int n ){

}
```

La fonction quartetToHexa (1)

La fonction **quartetToHexa** retourne un caractère égal au chiffre hexadécimal du quartet passé en paramètre.

quartet	car hexa
0	'0'
...	...
9	'9'
10	'a'
11	'b'
12	'c'
13	'd'
14	'e'
15	'f'

La fonction quartetToHexa (2)

quartetToHexa (5) retourne le caractère '5'

quartetToHexa (10) retourne le caractère 'a'

```
public static char quartetToHexa(int quartet ){

}
```

La fonction octetToHexa

La fonction **octetToHexa** retourne une chaîne de caractères égale au nombre hexadécimal de l'octet passé en paramètre.

```
octetToHexa ( 25 )
    retourne  la chaîne de caractères "19"

octetToHexa ( 11 )
    retourne  la chaîne de caractères "0b"
```

```
public static String octetToHexa(int octet ){


}
```

La fonction setBit

La fonction **setBit**

```
public static int setBit ( int b , int i ) {

}
```

retourne la valeur l'octet **b** avec le bit de rang **i** forcé à 1.

On suppose que

- la valeur de l'octet **b** est comprise entre 0 et 255
- la valeur du rang **i** est comprise entre 0 et 7

La fonction resetBit

La fonction **resetBit**

```
public static int resetBit ( int b , int i ) {

}
```

retourne la valeur l'octet **b** avec le bit de rang **i** forcé à 0.

On suppose que

- la valeur de l'octet **b** est comprise entre 0 et 255
- la valeur du rang **i** est comprise entre 0 et 7

La fonction compBit

La fonction **compBit**

```
public static int compBit ( int b , int i ) {

}
```

retourne la valeur l'octet **b** avec le bit de rang **i** complémenté.

On suppose que

- la valeur de l'octet **b** est comprise entre 0 et 255
- la valeur du rang **i** est comprise entre 0 et 7