Les opérateurs de masque

ET binaire (&) ou AND			
0	0	**0**	On constate pour le **ET** bit à bit : ➢ le **0** est l'élément **absorbant** ➢ **le 1 est l'élément neutre**
0	1	**0**	
1	**0**	**0**	
1	**1**	**1**	

OU binaire (|) ou OR			
0	**0**	**0**	On constate pour le **OU** bit à bit : ➢ le **0** est l'élément **neutre** ➢ le **1** est l'élément **absorbant**
0	**1**	**1**	
1	0	**1**	
1	1	**1**	

Les autres opérateurs très utilisés :

- variable **>> n** : décalage à **droite** de ***n*** bits du contenu de la variable
- variable **<< n** : décalage à **gauche** de ***n*** bits du contenu de la variable
- **~**variable : **inversion bit à bit** du contenu de la variable

OU exclusif (^) ou XOR			
0	**0**	**0**	On constate pour le **OU EXCLUSIF** : ➢ le **0** est l'élément **neutre** ➢ le **1** est l'élément **inverseur**
0	**1**	**1**	
1	**0**	**1**	
1	**1**	**0**	

Remarques : le OU EXCLUSIF a des propriétés remarquables ...

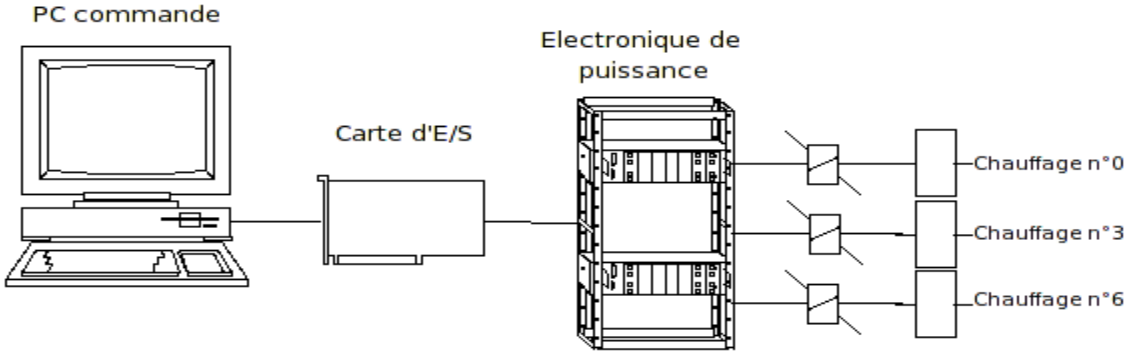
1 . **`variable = variable^variable; // variable est mise à 0 !`**

2 . Cryptage et décryptage simple XOR

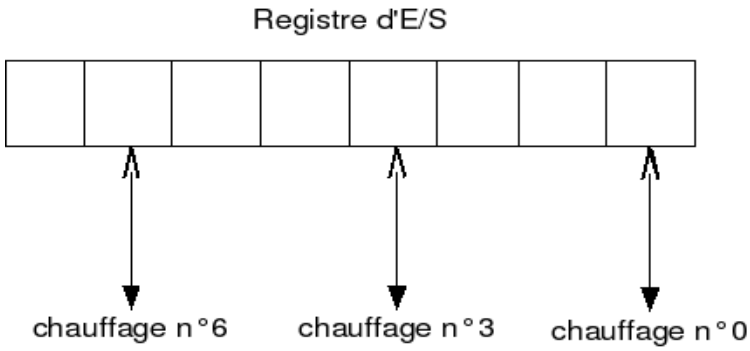
```
#include <stdio.h>

int main()
{
   unsigned char data, key, dataCrypt, dataDecrypt;

   data = 'A'; // la data à crypter
   key  = 'X'; // la clé

   // cryptage XOR
   dataCrypt = data ^ key;
   printf("data=%c (0x%2X) -> dataCrypt=0x%2X\n", data, data, dataCrypt);

   // décryptage XOR
   dataDecrypt = dataCrypt ^ key;
   printf("dataCrypt=0x%2X -> dataDecrypt=%c (0x%2X)\n", dataCrypt, dataDecrypt,
dataDecrypt);
   return 0;
}
```

Exercices sur les masques

Soit un **octet** dont les 8 bits représentent la commande individuelle d'éléments chauffants (8 bits = 8 chauffages). Par exemple, si le bit 0 est à 1 alors le chauffage n°0 est allumé et si le bit 0 est à 0 alors le chauffage n°0 est éteint etc ...



Cette installation est simplement vue par le programmeur en informatique industrielle :



La difficulté est de commander individuellement chaque chauffage (donc sans toucher à l'état des autres chauffages).

1 . Donner une déclaration en C de la variable **commandeChauffages**.
Réponse :

2 . Ecrire la commande pour **activer seulement le chauffage n°3**.
Réponse :

opérateur : ___

7	6	5	4	3	2	1	0	
X	***X***	***X***	***X***	***0***	***X***	***X***	***X***	***= état précédent***
								= masque (0x__)
X	X	X	X	1	X	X	X	= activation du chauffage n°3

En langage C :

3 . Ecrire la commande pour **éteindre seulement le chauffage n°6**.
Réponse :

	7	6	5	4	3	2	1	0	
	X	***1***	***X***	***X***	***X***	***X***	***X***	***X***	***= état précédent***
opérateur : _ _ _									= masque (0x_ _)
	X	0	X	X	X	X	X	X	= activation du chauffage n°6

En langage C :

Remarque : l'écriture de valeur sans signification dans un programme est à proscrire pour des raisons de *debugage*, de maintenance et d'évolution (par exemple 0xFE ??)

4 . Suite à la remarque précédente, proposer une (ou des) solutions pour éviter d'utiliser des valeurs sans signification.
Réponse :

5 . Que réalise ces lignes de codes ? Compléter le tableau ci-dessous.
Réponse :

```
#define ARRET_CHAUFFAGES      0x00
#define CHAUFFAGE_3           0x08
#define CHAUFFAGE_2           0x04
#define CHAUFFAGE_1           0x02
#define CHAUFFAGE_0           0x01
```

	7	6	5	4	3	2	1	0	en hexa	*Action*
`commandeChauffages = ARRET_CHAUFFAGES;`	***0***	***0***	***0***	***0***	***0***	***0***	***0***	***0***	***= 0x00***	***état initial à l'arrêt***
`commandeChauffages |= CHAUFFAGE_3;`	0	0	0	0					= 0x	
`commandeChauffages &= ~CHAUFFAGE_3;`	0	0	0	0					= 0x	
`commandeChauffages |= (CHAUFFAGE_1 | CHAUFFAGE_0);`	0	0	0	0					= 0x	
`commandeChauffages &= ~(CHAUFFAGE_1 | CHAUFFAGE_0);`	0	0	0	0					= 0x	
`commandeChauffages |= CHAUFFAGE_2;`	0	0	0	0					= 0x	

On désire maintenant connaître l'état des chauffages individuellement.

6 . Donner une déclaration en C de la variable **etatChauffages**.
Réponse :

7 . Afficher l'état logique du **chauffage n°2**.
Réponse :

	7	6	5	4	3	2	1	0	
	X	X	X	X	X	0/1	X	X	= *état des chauffages*
opérateur : ___									= masque (0x__)
	0	0	0	0	0	0/1	0	0	= état du chauffage n°2
opérateur : ___									
	0	0	0	0	0	0	0	0/1	= état booléen du chauffage n°2

En langage C :

```
if( etatChauffage            == 1 )
   printf("chauffage n°2 : allumé\n");
else
   printf("chauffage n°2 : éteint\n");
```

8 . Ecrire la fonction **fabriquerCommandeChauffage()** qui **reçoit en paramètres** le numéro de chauffage (0 à 7) et l'action à réaliser (0 ou 1) et qui **retourne** la valeur de la commande.
Réponse :

```
#define ARRET     0
#define MARCHE    1
#define OFF       0
#define ON        1
typedef unsigned char BYTE; // définition du type BYTE (c'est un unsigned char)

.............. fabriquerCommandeChauffage(................., .................)
{




}
```

9 . Dans la fonction suivante, déterminer le rôle et l'utilisation des deux paramètres **choix** et **cmd** :

```
typedef unsigned char BYTE; // définition du type BYTE (c'est un unsigned char)

BYTE fabriquerCommande2Chauffage(BYTE choix, BYTE cmd)
{
   BYTE etat, commande;

   etat = lire(); // cette fonction peut être simulée pour des tests

   commande = etat & ~choix ;
   commande = commande | (choix & cmd);

   return commande;
}
```

Réponse :

	7	6	5	4	3	2	1	0	
etat	X	X	X	X	X	X	X	X	
choix									0x48
~choix									
etat & ~choix									
cmd									0x08
choix & cmd									
commande | (choix & cmd)									

choix :

cmd :

10 . Comparer les deux fonctions réalisées **fabriquerCommandeChauffage** et **fabriquerCommande2Chauffage**.

Réponse :