Liste d'exercices 4 : les nombres entiers signés et non signés

Notions à considérer pour pouvoir faire les exercices

La matière du cours se trouve dans le chapitre 3 dans les notes de cours. Relire attentivement les notions à considérer présentées dans les listes d'exercices 2 et 3.

Priorité des opérateurs:

Priorités	*Opérateurs en C*	*Instructions en assembleur*
++++++++	`()`	
+++++++	`~, -, (changement genre)`	`not, neg, movzx→movsx/movsx→movzx`
++++++	`*, /, %`	`imul/mul, idiv/div, idiv/div`
+++++	`+, -`	`add, sub`
++++	`&`	`and`
+++	`^`	`xor`
++	`|`	`or`
+	`=`	`mov`

Instructions logiques:

- *AND opd, ops* stocke dans l'opérande *opd* le résultat de l'opération et logique entre l'opérande *ops* et l'opérande *opd*. Cette instruction est utilisée pour mettre à 0 certains bits d'un opérande sans changer la valeur des autres bits.

- *OR opd, ops* stocke dans l'opérande *opd* le résultat de l'opération ou logique entre l'opérande *ops* et l'opérande *opd.* Cette instruction est utilisée pour mettre à 1 certains bits d'un opérande sans changer la valeur des autres bits.

- *XOR opd, ops* stocke dans l'opérande *opd* le résultat de l'opération ou exclusif entre l'opérande *ops* et l'opérande *opd.*

- *NOT op* inverse les bits de l'opérande *op.*

Le tableau suivant montre comment fonctionnent les opérations logiques sur les variables à un bit A et B :

A	*B*	*A et B*	*A ou B*	*A ou exclusif B*	*non A*	*non B*
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Exercices

Exercice 1: écrivez la séquence d'instructions en assembleur qui correspond à chaque instruction d'affectation en langage C suivante :

```
int  i = -3;
short s = 4;
char  b = -2;
unsigned int i1 = 8;
unsigned short s1 = 4;
unsigned char b1 = 5;
```

- `i = i1 - (3 & b) - s;` `// i = 2`
- `s = -i - 3 + (unsigned char)b ^ (short)s1;` `// s = 250`
- `s1 = (i * 6 | s1) + 3 + b;` `// s1 = 65519`
- `b = b1 * s1 * i | i - 8;` `// b = -11`
- `s1 = i1 * 100 ^ 255 - b1 | (unsigned char)b;` `// s1 = 1022`

Exercice 2: à partir de chaque séquence d'instructions en assembleur donnée à la suite, retrouvez l'instruction d'affectation qui correspond en langage C :

- ```
  short s = 4;
  unsigned char b1 = 5;
  unsigned int  i1 = 8;

  int main()
  {
        _asm
        {
              mov   eax, i1
              movzx ebx, b1
              add   eax, ebx
              mov   edx, 0
              movzx ebx,s
              div   ebx
              mov   ebx, 8
              mul   ebx
              mov   i1, eax          // i1 = 24
        }

        return 0;
  }
  ```
  ```

- ```
  unsigned int   i = 35;
  unsigned short s = 8;
  unsigned char  b = 4;

  int main()
  {
        _asm
        {
              mov   eax, i
              mov   edx, 0
              mov   ebx, 3
              div   ebx
              not   eax
              mov   ecx, eax
              movzx eax, b
              mov   ebx, 232
              mul   ebx
              sub   ecx, eax
              mov   s, cx             // s = 64596
        }

        return 0;
  }
  ```

- ```
  unsigned char b = 5;
  short s = 0xbe15;
  int   i = -60000;

  int main()
  {
        _asm
        {
              movzx eax, b
              sub   eax, 0xffe1
              mov   ebx, 3
              mul   ebx
              mov   ebx, i
              sub   ebx, 8
              movsx ecx, s
              add   ebx, ecx
              add   eax, ebx
              mov   i, eax            // i = -273383
        }

        return 0;
  }
  ```