Liste d'exercices 2 : les nombres entiers (1 séance)

Notions à considérer pour pouvoir faire les exercices

Priorité des opérateurs du langage C :

Priorités	*Opérateurs en C*	*Instructions en assembleur*
+++++	`()`	
++++	`-`	`neg`
+++	`*, /, %`	`imul, idiv, idiv`
++	`+, -`	`add, sub`
+	`=`	`mov`

Un opérateur unaire est plus prioritaire qu'un opérateur binaire. Les parenthèses servent à changer la priorité par défaut d'opérateurs.

Instructions additionnelles sur entiers :

- *MOVSX opd, ops* : elle étend le nombre entier **signé** de l'opérande *ops* à la taille de l'opérande *opd*, puis elle copie ce nombre étendu dans *opd*. Concrètement, le processeur propage la valeur du bit de signe du nombre se trouvant dans *ops* vers tous les bits du/des octets ajoutés devant le nombre à étendre.

  Exemple :

```
char  b = 25;      // b est une variable en mémoire d'1 octet
int   i = -110;    // i est une variable en mémoire de 4 octets
```

Langage C	*Assembleur*
`i = b;`	`movsx eax, b` `mov  i, eax`

  On a étendu la valeur de la variable b à 4 octets, puis on a assigné cette valeur à la variable i.

- *NEG op :* elle change le signe de la valeur se trouvant dans *op.*

  Exemple :

```
char b = -3;
int  i;
```

Langage C	*Assembleur*

`i = -b;`	`movsx eax, b`
	`neg  eax`
	`mov  i, eax`

La valeur -3 de la variable b est étendue à 32 bits. Ensuite, le complémente à 2 de cette valeur est obtenu et copié dans la variable i.

- *CDQ* : elle étend le nombre entier de EAX à la paire EDX, EAX. Elle précède toujours l'instruction idiv, car elle sert à préparer le dividende.

- *IDIV op* : cette instruction sert à diviser un nombre entier par un autre. Le nombre entier signé de 64 bits formé par la concaténation des registres EDX et EAX est divisé par un nombre entier signé de 32 bits figurant dans l'opérande *op*. Après l'opération, EAX stocke le quotient et EDX stocke le reste de la division.

Exemple pour le modulo :

```
int  i, j;
```

Langage C	*Assembleur*

`i = i % j;`	`mov  eax, i`
	`cdq`
	`idiv j`
	`mov  i, edx`

On a préparé dans edx, eax le dividende sur 64 bits en utilisant cdq. Puis on a divisé ce dividende par la valeur de la variable j. Enfin, on a copié dans la variable i le reste de la division figurant dans le registre edx.

Exemple pour la division :

```
int  i, j;
```

Langage C	*Assembleur*

`i = i / j;`	`mov  eax, i`
	`cdq`
	`idiv j`
	`mov  i, eax`

On a préparé dans edx, eax le dividende sur 64 bits en utilisant cdq. Puis on a divisé ce dividende par la valeur de la variable j. Enfin, on a copié dans la variable i le quotient figurant dans le registre eax.

Exercices

Écrivez la séquence d'instructions en assembleur qui correspond à chaque instruction d'affectation en langage C suivante (en commentaire en vert figure la valeur à obtenir au final).

```
char  b = 25;    // b est une variable en mémoire d'1 octet avec 25 pour valeur
short s = 40;    // s est une variable en mémoire de 2 octets avec 40 pour valeur
int   i = -110;  // i est une variable en mémoire de 4 octets avec -110 pour valeur
```

- `i = -s + 2;` `// i = -38`
- `i = i + b + s;` `// i = -45`
- `i = i * b + s * i;` `// i = -7150`
- `i = -i * (b + s) * i;` `// i = -786500`
- `i = (i + b) * -(s + i);` `// i = -5950`
- `i = i / -s;` `// i = 2`
- `i = (i - s) % 31;` `// i = -26`
- `i = (i * b) / -(2 * b);` `// i = 55`
- `i = -(0x12 - i + b) / 5 - i * s / 2;` `// i = 2170`
- `s = i + -s * 18 - i % b;` `// s = -820`
- `s = -b - -(s / 3 * i + 800 / -i);` `// s = -1448`
- `s = -(b % 10) * +0xff - -(s / 3);` `// s = -1262`
- `i = -(s * 0xffe1) / -(b * -(s / 3));` `// i = -8062`
- `i = -(s * 0xffe1) / -(b * -(s / 3) + (3 - i)) / 2;` `// i = -6179`