Exercice 1 : Écrire un programme qui :

— stocke deux entiers dans la mémoire vive avec les labels **`val1`** et **`val2`**,

— échange ses deux valeurs.

Exercice 2 : Traduire les instructions suivantes en assembleur :

```
x = y + 42
if x == 42:
    z = 1
else:
    z = 2
```

Code 1 – Un code en Python

Exercice 3 :

1. Écrire puis exécuter le programme 2.

```
      MOV R0, #tab      //charge l'adresse du début du tableau
      LDR R1, longueur  //charge la longueur du tableau
      MOV R2, #0        //prépare un compteur
boucle:
      LDR R3, [R0]      //charge la valeur à l'adresse [R0] de la m
   émoire
      STR R3, .WriteUnsignedNum //affiche la valeur
      ADD R0, R0, #4    //passe à l'adresse suivante
      ADD R2, R2, #1    //augmente le compteur de 1
      CMP R2, R1        // vérifie si on est en fin de tableau
      BLT boucle
      HALT
longueur:10
tab:  13
      9
      10
      12
      51
      8
      71
      19
      3
      11
```

Code 2 – Réaliser une boucle

2. Certaines notions non encore abordées sont présentes dans ce programme :
   — Il répète un code plusieurs fois.
   — Le label **`tab`** peut être assimilé à un tableau de données.

   Prendre le temps de comprendre le fonctionnement du programme. Observer notamment le contenu de la mémoire vive.
3. Modifier le programme pour qu'il recherche et affiche dans la console la plus grande valeur de **`tab`** (on considère que le tableau ne contient que des entiers positifs ou nuls).

Exercice 4 :

Christophe Viroulaud

1. Que fait le programme 3 ?

```
1       MOV R0,#0
2       MOV R1,#10
3  ici:
4       CMP R1,#0
5       BEQ la
6       ADD R0,R0,R1
7       SUB R1,R1,#1
8       B ici
9  la:
10      HALT
```

Code 3 – Que fait ce programme?

2. Les microprocesseurs Intel utilisent le langage d'assemblage x86. Ce langage a huit registres : eax, ebx, ecx, edx, esi, edi, esp et ebp. La documentation (très incomplète) ci-dessous présente plusieurs instructions de ce langage :

Instructions	Utilisation	Exemple
mov destination, source	copie la source dans la destination	mov eax, 42
add destination, source	ajoute la source à la destination	add eax, 10
sub destination, source	soustraie la source à la destination	sub eax, 10
cmp source1, source2	effectue la soustraction source1-source2 et indique le résultat de comparaison dans des registres spéciaux (flag). Si le flag *zf* vaut 1 c'est que *source1* est égal à *source2*. Si le flag *sf* vaut 1 c'est que *source1* est inférieure à *source2*.	cmp eax, ebx
je adr	saute à l'adresse si le flag *zf* vaut 1. L'adresse est une étiquette ou label.	je label
jne adr	saute à l'adresse si *zf* vaut 0. L'adresse est une étiquette ou label.	jne label
jmp adr	saute à l'adresse. L'adresse est une étiquette ou label.	jmp label

Traduire le code 3 dans le langage assembleur x86.

Christophe Viroulaud