Les tableaux

Un tableau est un bloc, contigu de données en mémoire, dont tous les éléments sont d'un même type et d'une même taille.

Chaînes de caractères

Une chaîne de caractères est un tableau d'octets dont le dernier élément est 0.

Déclaration de tableaux

```
segment .data
; définit un tableau de 5 octets initialisés
; t1 1, 2 , . . , 5
t1:
        .byte 1 , 2 , 3 , 4 , 5

; définit un tableau de 7 mots doubles initialisés
t2:
        .double 0 , 0 , 0 , 0 , 0 , 0 , 0

; en utilisant .rept pour définir un tableau de 100 mots doubles initialisés à zéro
t3:
        .rept 100
        .long 0
        .endr
```

Les tableaux non initialisés sont définit dans le segment .bss

Accès aux éléments d'un tableau

```
array1:
        .long 50 , 40 , 30 , 200 , 100
```

$array1 = adresse du premier élément du tableau
array1 = valeur du premier élément du tableau
$array1 + 4 = adresse du second élément
$array1 + 8 = adresse du troisième élément

La pile

De nombreux processeurs ont un support intégré pour une pile.
Une pile est une liste Last-In First-Out (LIFO) : dernier entré, premier sorti.
push ***reg*** ajoute une donnée sur la pile de taille quadruple word (64 bits). Cette donnée se trouve au sommet de la pile.
pop ***reg*** retire la donnée de taille quadruple word qui se trouve au sommet de la pile.
Le registre rsp (esp sur les machines 32 bits) contient l'adresse de la donnée qui se trouve au sommet de la pile.
push reg décrémente rsp de 8 et pop incrémente rsp de 8.

Pourquoi utiliser une pile?

- Sauvegarder les valeurs des registres
- Passage des paramètres lors de l'appel de sous programme
- Stockage de variables locales
- Stockage des adresses de retour (lors des appels de fonctions)

Exercice1: fusion de deux tableaux triés
Exercice2: tri par sélection

Tri par sélection: On parcourt le tableau à trier du début à la fin. Au moment où on considère le *i*-ème élément, les éléments qui le précèdent sont déjà triés. Pour faire l'analogie avec l'exemple du jeu de cartes, lorsqu'on est à la *i*-ème étape du parcours, le *i*-ème élément est la carte saisie, les éléments précédents sont la main triée et les éléments suivants correspondent aux cartes encore mélangées sur la table.
L'objectif d'une étape est d'insérer le *i*-ème élément à sa place parmi ceux qui précèdent. Il faut pour cela trouver où l'élément doit être inséré en le comparant aux autres, puis décaler les éléments afin de pouvoir effectuer l'insertion. En pratique, ces deux actions sont fréquemment effectuées en une passe, qui consiste à faire « remonter » l'élément au fur et à mesure jusqu'à rencontrer un élément plus petit.
Voici une description en pseudo-code de l'algorithme présenté. Les éléments du tableau *T* sont numérotés de 0 à *n-1*.

```
procédure tri_insertion(tableau T, entier n)
  pour i de 1 à n-1
     x ← T[i]
     j ← i
     tant que j > 0 et T[j - 1] > x
        T[j] ← T[j - 1]
        j ← j - 1
     fin tant que
     T[j] ← x
  fin pour
fin procédure
```