INF 4170 – Architecture des ordinateurs

TP1

- À faire en groupe de **deux** étudiants.
- Remise : Le **15 novembre** 2018 à **23:55** au plus tard
- Il y aura une pénalité de **10%** par jour de retard.

Programmation en assembleur MIPS

Un tas (monceau) est une structure de données représentant un arbre binaire presque complet (seulement la dernière couche du côté droite pourrait être incomplète) qui permet de retrouver la plus grande valeur ou la plus petite valeur de tous les éléments du tas de manière efficace.
Considérons un tas MIN. Dans ce type de tas les clés sont ordonnés selon la propriété de tas : la clé d'un nœud parent doit être plus petit que les clés de ses enfants. Comme un tas possède une structure très régulière, on l'implémente avec un tableau. Dans les prochaines 2 numéros vous allez implémenter un trie «Heap sort » qui permettra de trier un tableau des éléments en ordre décroissant.

1) Écrire un code MIPS qui initialise un tableau de **`n`** valeurs. Ces valeurs doivent être placées à partir de l'adresse 0x10010000 (région de **`heap`**). D'abord, vous devez chercher le paramètre **`n`** saisi par un utilisateur qui représente le nombre d'éléments du tableau à traiter. Ensuite, cherchez les éléments du tableau et stockez les à partir de l'adresse spécifiée.
   Utilisez l'appel système correspondant pour pouvoir saisir toutes les valeurs nécessaires.

   Concevoir les fonctions **`swap, getLeftChildIndex`** et **`getRightChildIndex`** ainsi que la fonction **`main`** qui fera les appels à ces trois fonctions en les testant.
   La fonction **`swap`** échange le contenu de deux éléments du tableau avec les indices **`i`** et **`j`** respectivement. Les fonctions **`getLeftChildIndex`** et **`getRightChildIndex`** retournent les indices des enfants gauche et droite dans la structure de données TAS.

```
void swap(int i, int j)
{
      int temp = a[i];
      a[i] = a[j];
      a[j] = temp;
}
int getLeftChildIndex(int index)
{
    return 2 * index + 1;
}
int getRightChildIndex(int index)
{
      return 2 * index + 2;
}
```

2. Rajouter les implémentations de deux fonctions

```
void fixHeap(int rootIndex, int lastIndex);
void sort(int [], int par_taille);
```

qui compléteront l'implémentation de l'algorithme **HeapSort**. Écrire une fonction **main** qui fait l'appel au **HeapSort** pour trier un tableau de **n** éléments. Incorporer votre code d'initialisation d'un tableau du numéro 1. Le code Java de **HeapSort**, de la fonction **main** en Java ainsi que les notes de cours Programmation II, qui traitent ce sujet sont fournis sur le site du cours.

Directives.

Dans toutes les questions, il ne faut pas utiliser ni la multiplication, ni la division, ni la virgule flottante. Il faut obligatoirement respecter toutes les conventions MIPS.

Imprimer une valeur stockée dans le registre `$t0` :

```
            li  $v0, 1   # service 1 imprime un entier
# charger la valeur à imprimer dans le registre $a0.
# Notre valeur se trouve dans $t0

            add $a0, $t0, $zero
            syscall            # faire appel du service
```

Imprimer une chaîne se trouvant à l'adresse msg

```
     la   $a0, msg
     li   $v0, 4    # Print string (Code 4 de syscall)
     syscall        # faire appel du service
```

À remettre :

2 fichiers *.asm pour les numéros 1 et 2

Barème de correction :

Bonne fonctionnement :
QI : 30 pts
QII : 40 pts
Utilisation des conventions : 5*2 = 10
Utilisation correcte de la pile : 10
Respect des directives, propreté, présentation, commentaires significatifs dans les programmes, etc. : 10

Total : 100

Bon travail!