INF1600

Travail pratique 3

Lien C++ et Assembleur

Département de Génie Informatique et Génie Logiciel
Polytechnique Montréal

1 Introduction et sommaire

Ce travail pratique a pour but d'approfondir vos connaissances sur l'assembleur IA-32, toujours selon la syntaxe AT&T. Il sera question dans ce travail pratique d'écrire des programmes impliquant la notion de récursivité. Par la suite, vous implémenterez des méthodes de classes dont le comportement attendu vous est fourni sous forme de code C++. La dernière partie de ce travail pratique abordera finalement la notion de classe Parent et classe Enfant.

1.1 Remise

Voici les détails concernant la remise de ce travail pratique :

- Méthode : sur Moodle, une seule remise par équipe, incluant un **rapport PDF** et les **fichiers sources** que vous modifiez en un seul fichier compressé. Une pénalité de 0,5 pt sera appliquée si le rapport est remis sous un un format différent.
- Format du rapport PDF : Incluez une page titre où doivent figurer les noms et matricules des deux membres de l'équipe, votre groupe de laboratoire, le nom et le sigle du cours, la date de remise et le nom de l'École. Dans une seconde page, incluez le barème de la section 1.2. Finalement, pensez à incluez les réponses aux questions et des captures d'écran si requis.
- Format des fichiers sources : Modifiez les fichiers assembleurs demandés dans les dossiers désignés à chaque question, puis compresser le tout en un seul fichier source <.Zip> que vous devez nommer comme suit :
  `<matricule1>-<matricule2>-<tp2>.<Zip>`

Attention :

L'équipe de deux que vous avez formé pour le TP1 est la même pour ce TP et la suite des TPs de cette session.

1.2 Barème

Les travaux pratiques 1 à 5 sont notés sur 4 points chacun, pour un total de 20/20. Le TP3 est noté selon le barème suivant. Reproduisez ce tableau dans le document PDF que vous remettrez.

TP 3			**/4,00**
Factorielle en Assembleur			/1,00
	Q1	/0,25	
	Q2	/0,5	
	Q3	/0,25	
Tours de Hanoi et récursivité			/1,00
	Q1	/1,00	
Crise du logement			/2,00
Partie 1			/1,00
	Q1	/0,5	
	Q2	/0,5	
Partie 2			/1,00
	Q3	/0,25	
	Q4	/0,75	

2 Factorielle en Assembleur

Pour ce premier exercice, il sera question d'implémenter une fonction permettant de calculer la factorielle d'un nombre. Pour rappel, on définit la factorielle d'un nombre n :

$$n! = \prod_{1 \leq i \leq n} i = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$$

Par exemple :

$$1! = 1$$

$$2! = 2 \times 1$$

$$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 3\,628\,800$$

Avec la **particularité** :

$$0! = 1$$

Pour exécuter votre code assembleur, nous avons fait appel aux fonctions correspondantes dans le fichier `main.c`. **Vous ne devez pas modifier ce fichier**. Pour lancer le programme, exécutez les commandes suivantes dans le terminal, en vous trouvant dans le répertoire courant de l'exercice :

```
$ make clean
$ make
$ ./factorielle
```

2.1 Partie 1 : Factorielle itérative

Q1/ À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `factorielle_s_iter.s` qui doit calculer et retourner la valeur correspondante à la factorielle de la valeur passée en paramètre, de façon **itérative**. La fonction correspondante en C se trouve dans le fichier `main.c`. Vous pouvez vous inspirer de cette fonction.

Lorsque vous lancez le programme, la version en C fournie est exécutée en simultané avec votre implémentation en Assembleur. Comparez les valeurs affichées dans le terminal pour vérifier le bon fonctionnement de votre implémentation.

2.2 Partie 2 : Factorielle récursive

Q2/ Décommentez les lignes correspondantes aux appels des fonctions récursives dans le fichier `main.c` pour exécuter votre code et le code fourni.

À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `factorielle_s_rec.s` qui doit calculer et retourner la valeur correspondante à la factorielle de la valeur passée en paramètre, de façon **récursive**. La fonction correspondante en C se trouve dans le fichier `main.c`. **Une note de 0 sera attribuée à cette question si l'implémentation n'est pas récursive**. Comparez les valeurs affichées dans le terminal pour vérifier le bon fonctionnement de votre implémentation.

Q3/ Dans le fichier `main.c`, remplacez la valeur de `nombre_valeurs` par 500. Quel comportement observez-vous ? Justifier le résultat produit par le programme, en vérifiant que le comportement est identique entre les implémentations en Assembleur et les implémentations en C.

3 Tours de Hanoi et récursivité

Soit le « casse-tête » des Tours de Hanoi, inventé par le Français Édouard Lucas (le même Lucas dont les travaux ont été abordés dans le travail pratique précédent).

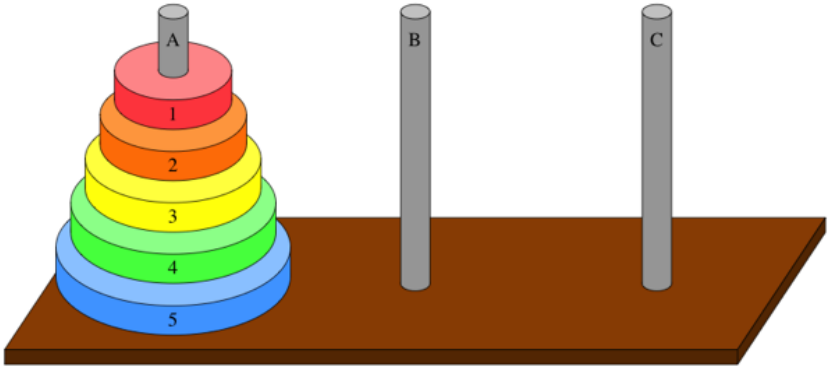

Figure 1 - Illustration des Tours de Hanoi, source: Khan Academy

Le problème des tours de Hanoi consiste à transférer *n* disques (les disques 1, 2, 3, 4, 5 dans cet exemple) du point A au point C en utilisant le point B comme point d'arrêt intermédiaire, tout en respectant les contraintes suivantes :

- Un seul disque peut être déplacé à la fois
- Un disque ne peut être placé que sur un autre disque de plus grande taille ou sur un point vide

Une solution récursive à ce problème se formule comme suit :

1. Déplacer les $n - 1$ disques du haut du point A au point B en utilisant C comme point intermédiaire ;
2. Déplacer le disque n du bas du point A au point C ;
3. Déplacer les $n - 1$ disques du point B au point C en utilisant A comme point intermédiaire.

Pour cet exercice, vous devez implémenter en Assembleur une fonction récursive permettant de résoudre le problème de Hanoi. Une référence implémentée en `C` vous est donnée.

Pour exécuter votre code assembleur, nous avons fait appel aux fonctions correspondantes dans le fichier `main.c`. **Vous ne devez pas modifier ce fichier**. Pour lancer le programme, exécutez les commandes suivantes dans le terminal, en vous trouvant dans le répertoire courant de l'exercice :

```
$ make clean
$ make
$ ./hanoi
```

Q1/ À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `hanoi_s_rec.s` qui implémenter **un comportement identique** à la fonction équivalente en C `hanoi_c` se trouvant dans le fichier `main.c`. **Une note de 0 sera attribuée à cette question si l'implémentation n'est pas récursive**. Comparez les valeurs affichées dans le terminal pour vérifier le bon fonctionnement de votre implémentation.

4 Crise du logement

Dans le cadre de cet exercice, il sera question d'implémenter des fonctions permettant de calculer la surface au sol de bâtiments en fonction du nombre d'étage et de la superficie de chaque étage et d'évaluer le prix des bâtiments en fonction de leur superficie, de leur zone de localisation (Ville, Campagne, Zone Industrielle) et du prix du mètre carré **avant la crise du logement**.

Pour exécuter votre code assembleur, nous avons fait appel aux fonctions correspondantes dans le fichier `main.c`. **Vous ne devez pas modifier ce fichier**.

Pour lancer le programme de la première partie, exécutez les commandes suivantes dans le terminal, en vous trouvant dans le répertoire courant de cette première partie:

```
$ make clean
$ make
$ ./batiment
```

Pour lancer le programme de la seconde partie, exécutez les commandes suivantes dans le terminal, en vous trouvant dans le répertoire courant de cette première partie:

```
$ make clean
$ make
$ ./immeuble
```

4.1 Partie 1 : Bâtiment général

Dans cette partie, il est question d'évaluer la surface au sol et le prix d'un bâtiment en général, sans prendre en compte le type de bâtiment.

Q1/ À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `calculerSurfaceSolAsm.s` qui doit implémenter **un comportement identique** à la fonction correspondante en C++ `Batiment::calculerSurfaceSol()` se trouvant dans le fichier `batiment.cpp`.

Q2/ À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `calculerPrixAsm.s` qui doit implémenter **un comportement identique** à la fonction correspondante en C++ `Batiment:: calculerPrix(…)` se trouvant dans le fichier `batiment.cpp`. Votre programme doit appeler la méthode `calculerSurfaceSolAsm()`.

4.2 Partie 2 : Immeuble

Dans cette partie, il est question d'évaluer la surface au sol et le prix d'un immeuble. **La valeur d'un immeuble dépend, en plus de la surface au sol, du nombre et du type d'appartement qu'il englobe**.

Commencez par copier les deux programmes (`calculerSurfaceSolAsm.s, calculerPrixAsm.s`) écrits précédemment dans le répertoire de cette partie de l'exercice.

Q3/ Corrigez les deux programmes copiés pour qu'ils puissent fonctionner dans ce nouveau contexte.

Q4/ À l'aide du jeu d'instructions IA-32, complétez le programme assembleur `calculerPrixAsmImmeuble.s` qui doit implémenter **un comportement identique** à la fonction correspondante en C++ `Immeuble:: calculerPrix()` se trouvant dans le fichier `batiment.cpp`.