Travail pratique #1

IFT-1215

February 13, 2015

¡¡ Dû le dimanche 8 mars à 23h59 !!

1 Survol

Ce TP a pour but de vous familiariser avec le langage d'assemblage et le langage machine.
Les étapes sont les suivantes:

1. Parfaire sa connaissance du LMC.
2. Lire et comprendre cette donnée.
3. Écrire le code.
4. Écrire un rapport. Il doit décrire **votre** expérience pendant les points précédents: problèmes rencontrés, surprises, choix que vous avez dû faire, options que vous avez sciemment rejetées, etc... Le rapport ne doit pas excéder 4 pages.

Ce travail est à faire en groupes de 2 étudiants. Le rapport doit être écrit en LaTeX (compilable sur `frontal.iro`). Le rapport et le code sont à remettre par remise électronique avant la date indiquée. Aucun retard ne sera accepté. Indiquez clairement vos noms au début de chaque fichier.

Si un étudiant préfère travailler seul, libre à lui, mais l'évaluation de son travail n'en tiendra pas compte. Si un étudiant ne trouve pas de partenaire, même après avoir annoncé sa disponibilité dans le forum de discussion, il doit me contacter au plus vite. Des groupes de 3 ou plus sont **exclus**.

2 Maintenir des nombres triés

Le travail consiste à implanter un programme qui lit deux séquences de nombres triés par ordre croissants, et renvoie la combinaison de ces deux séquences, elle aussi triée par ordre croissant.

Le programme lira (avec des instructions IN) une première séquence de nombres, où 0 sera utilisé pour indiquer la fin de la séquence. Il lira ensuite une deuxième séquence, de la même manière. On présume que chacune de ces deux séquences aura la propriété d'être triée par ordre croissant.

Finalement, il renvoie (avec des instructions OUT) une nouvelle séquence qui combine ces deux séquences tout en maintenant l'ordre croissant des éléments.

I.e. Si l'utilisateur entre les nombre 1, 3, 5, 0 puis 1, 2, 4, 0 la machine devrait répondre avec 1, 1, 2, 3, 4, 5.

À noter que le "Recent output" du simulateur LMC qui vous est fourni montre les résultats les plus récent à gauche et les plus anciens à droite, donc dans l'exemple ci-dessus il devrait terminer en indiquant `(005 004 003 003 001 001)`.

Pour ce programme, vous allez devoir stocker les nombres dans un *tableau*. Vu que les instructions LMC ne peuvent accéder qu'à des adresses fixes, il vous faudra utiliser du *self-modifying code*.

3 Remise

Il faudra remettre électroniquement, via la page Moodle (aussi connu sous le nom de StudiUM) du cours, 2 fichiers: `rapport.tex`, `sortmerge.elmc`. SVP utiliser **exactement** ces noms de fichiers. Indiquer clairement vos noms au début de chaque fichier.

4 Barème

Les 23 points en jeux seront divisés comme suit:

- 16 points pour le code.
- 7 points pour le rapport.

Le rapport doit clairement expliquer les choix que *vous* avez faits, ainsi que les *limitations* de votre code. Il peut aussi documenter les choix que vous avez écartés, en expliquant pourquoi.

Les points sur le code seront pour moitié basés sur de simples tests pour vérifier le comportement de vos programmes et pour moitié basé sur une revue manuelle de code. Les critères par ordre d'importance:

- Le code doit être correct dans un maximum de cas, même les plus tordus.
- Le code doit être *bien* commenté, i.e. ni trop, ni trop peu; expliquer ce que le code fait (et ne fait pas), sans le paraphraser.
- Le code doit suivre les conventions stylistiques habituelles: indentation standard, ne pas dépasser 80 colonnes, les commentaires sont traités avec dignité (e.g. ils ont droit à une majuscule au début et une ponctuation à la fin), ...
- Si une partie du comportement n'est pas spécifiée dans la donnée, il faut le mentionner dans le code et/ou le rapport et expliquer quel choix vous avez fait et pourquoi.
- *Faute avouée est à moitié pardonnée*: si votre code s'écarte du comportement stipulé par la donnée, il est préférable de le mentionner et expliquer dans le rapport, pour bien montrer que vous en êtes au moins conscient.