TD 5 - PROGRAMMATION EN ASSEMBLEUR

Arithmétique, appels de procédure, récursivité.
Page web : `http://www-rocq.inria.fr/~acohen/teach/archi.html`

Exercice 5.1 - Arithmétique multi-précision

Sur 16 bits, la valeur d'un nombre entier positif est comprise entre 0 et $2^{16} - 1$. Pour manipuler de grandes valeurs entières, il est nécessaire de créer des routines reproduisant les opérations arithmétiques sur de grands nombres, codés sur plusieurs mots. Dans cet exercice, on implémentera ces routines et on pourra ensuite les illustrer sur le calcul de la factorielle d'un entier.

On utilise deux nouveaux programmes pour faciliter l'assemblage et l'exécution sur le processeur LC-2.

- Script shell `~cohen/bin/lc2asm`.
  À utiliser pour appeler l'assembleur LC-2 (éviter de lancer directement l'assembleur).
  Utilisation :
  `> lc2asm fichier.asm fichier`
  - Produit `fichier.obj` : code "objet" exécutable sur le simulateur "logiciel" du LC-2 (voir ci-dessous, à ne pas confondre les circuits *DigLog*).
  - Produit `fichier.lst` : mise en parallèle des instructions, étiquettes, code machine binaire, code machine hexadécimal et adresses des instructions.
  - Produit `fichier.hex` : utilisé pour charger les composants SRAM8K de *DigLog* (voir ci-dessous).
  - Et autres extensions (`.bin`, `.log`, `.sym`) inutiles ici.

  Attention aux messages d'erreur (pas toujours très explicites).
- Script shell `~cohen/bin/xlc2sim`.
  Simulateur "logiciel" du LC-2, beaucoup plus rapide — et un peu plus convivial — que les circuits *DigLog*.
  Utilisation :
  `> xlc2sim`
  Suivre les menus du simulateur.

Question 5.1.1

Écrire en assembleur une fonction (appelée par `JSR`) qui ajoute `R0` à `R1` (deux entiers positifs en binaire sur 16 bits), puis range la somme sur 16 bits dans `R0` et range la retenue dans `R1`.

Question 5.1.2

Écrire une fonction réalisant la multiplication (non signée) de `R0` par `R1` avec un résultat sur 32 bits, en rangeant les bits de poids faible du résultat dans `R0` et les bits de poids fort dans `R1`.

Question 5.1.3

Un nombre entier *positif N* est représenté en *multi-précision* de la manière suivante : à une adresse donnée en mémoire, la représentation binaire de *N* sur 16*m* bits est codée sur *m* mots consécutifs, en commençant par les bits de poids fort (*big endian*). En pratique, on supposera que *m* est fixé à la valeur 16.

Écrire une fonction effectuant la multiplication du registre `R0` (entier sur 16 bits) par un entier sur 16*m* bits, avec un résultat sur 16*m* bits. On supposera que `R1` indique l'adresse du mot de poids fort de l'entier sur 16*m* bits, et que le résultat est rangé en commençant à l'adresse indiquée par `R2` (*m* mots doivent être préalablement réservés à cette adresse).

Question 5.1.4

Écrire un programme utilisant une procédure récursive pour calculer — en multi-précision — la factorielle de l'entier sur 16 bits contenu dans une variable nommée `Argument` ; le résultat sera rangé en mémoire dans le tableau de *m* mots nommé `Resultat`.

Vérifier par exemple que $17! = \texttt{x0001 437E EECD 8000}$ en hexadécimal (c'est-à-dire 355 687 428 096 000 en base 10).