Circuits et Architecture (CA7)
TD n° 8 : Assembleur LC-3

1 Rudiments de programmation LC-3

On va utiliser un sous-ensemble de l'assembleur LC-3 que vous avez vu en cours. Il est composé des instructions `NOT`, `ADD`, `AND`, `BR`, `JMP`, `LD`, `LDR`, `STR`.

Exercice 1 – *Initialisation à zéro*

Comment initialiser un registre à la valeur zéro ? Donner l'instruction assembleur puis son codage hexadécimal.

Exercice 2 – *Initialisation à n quelconque*

1. Comment mettre une constante, par exemple 2020, dans un registre, par exemple R0 ? Même question sans utiliser de chargement. Donner la ou les instructions assembleur.
2. Avec la technique précédente qui n'utilise pas de chargement, combien d'instructions sont nécessaires pour une constante arbitraire sur 16 bits ?

Exercice 3 – *Encodage de `NOP`*

Avec les instructions dont on dispose, donner toutes les façons de coder un `NOP`, c'est-à-dire une instruction sans effet.

Exercice 4 – *Soustraction*

Programmer une séquence d'instructions qui implémente R2 $\leftarrow$ R0 $-$ R1.

Exercice 5 – *Suite de Fibonacci*

Programmer le calcul des 12 premiers termes de la suite de Fibonacci grâce à une boucle.

Exercice 6 – *Somme tableau*

Programmer le calcul de la somme des éléments d'un tableau. L'adresse du début du tableau est stockée dans R0 et la somme doit être stockée dans R1. On supposera qu'on peut modifier tous les registres. Programmer la somme d'un tableau deux fois, en supposant :

1. d'abord, que la fin du tableau est marquée par un octet nul ;
2. puis, que sa taille est stockée dans R2.

2 Changements de contexte

On désire maintenant implémenter le changement de contexte, afin de pouvoir faire des appels de fonctions et, très schématiquement, de la programmation multitâches.

Dans ce but, nous allons implémenter une *pile* en mémoire. Par convention, durant cette séance, $\mathtt{Mem}[0]$ va contenir le sommet de pile, c'est-à-dire l'adresse de sa première case vide. La première case de la pile est $\mathtt{Mem}[1]$, la seconde $\mathtt{Mem}[2]$, et ainsi de suite jusqu'à la dernière $\mathtt{Mem}[\mathtt{Mem}[0]-1]$. La pile croît donc vers le haut.

Exercice 7 – *Sauvegarde du contexte sur la pile*

1. Programmer la sauvegarde du contexte, c'est-à-dire des registres R1-R7, sur la pile.

2. Pourquoi ne peut-on pas sauvegarder tous les 8 registres ?

Exercice 8 – *Restauration du contexte*

Programmer la restauration du contexte : les registres R1-R7 doivent reprendre leurs valeurs avant la sauvegarde, tout comme le pointeur de pile.