IFT 1215 H21 - Introduction aux systèmes informatiques
Démonstration Semaine 8 - Solutions

Blanche Mongeon

18 mars 2021

Problème 6.5

Prendre l'exemple vu précédemment du programme LMC qui lit deux nombres et les additionne, et au lieu de stocker le premier nombre dans la case mémoire 99 supposons qu'on le stocke à la case mémoire 00. Le résultat de l'exécution du programme serait-il affecté ? Et si on exécute le programme une deuxième fois ? Pourquoi ?

Le résultat ne sera pas affecté lors de la première exécution. En effet, puisque la case mémoire 00 va contenir le 1er nombre, lorsqu'on va l'exécuter la première fois, alors à l'étape 03, on va bel et bien additionner le premier nombre au deuxième. Cependant, lors de la deuxième exécution, le programme n'aura pas le résultat escompté. En effet, après la première exécution, la première instruction sera devenue la valeur du premier nombre entré. Lors de la seconde exécution, l'instruction IN ne sera donc pas reconnue.

Problème 6.6

Écrire un programme en assembleur LMC qui lit trois nombres et renvoie le plus grand des trois.

```
#programme LMC qui lit 3 nombres et renvoie le plus
#grand des 3

        IN
        STO     MAX
boucle  LDA     N
        SUB     UN
        STO     N #N contient combien de
                  #nbres il reste à lire
        BRZ     FIN
        IN
        SUB     MAX
        BRP     NEWMAX
        BR      boucle

NEWMAX  ADD     MAX
        STO     MAX
        BR      boucle

FIN     LDA     MAX
        OUT
        HLT

MAX     DAT     00
N       DAT     03
UN      DAT     01
```

Figure 1 – Programme en LMC qui lit 3 nombres et renvoie le plus grand des trois

Problème 6.7 v1

Écrire un programme en assembleur LMC qui lit un nombre arbitraire de nombres et renvoie le plus grand de ces nombres. Utiliser le nombre 0 pour indiquer la fin de la séquence de nombres.

```
#programme LMC qui lit un nombre arbitraire de nombre et
#renvoie le plus grand

FIRST   IN
        STO     MAX
        BRZ     FIN
boucle  IN
        BRZ     FIN #le nombre 0 indique la fin
        SUB     MAX
        BRP     NEWMAX
        BR      boucle

NEWMAX  ADD     MAX
        STO     MAX
        BR      boucle

FIN     LDA     MAX
        OUT
        HLT

MAX     DAT     000
```

FIGURE 2 – Programme qui lit un nombre arbitraire de nombres et renvoie le plus grand de ces nombres

Problème 6.7 v2

Comme le problème 6.7 *v1, mais cette fois-ci le programme doit renvoyer les deux nombres les plus grands.*

```
#programme LMC qui lit un nombre arbitraire de nombre et
#renvoie les deux plus grands
FIRST   IN
        STO     MAX1
        BRZ     END
2ND     IN
        BRZ     END
        STO     N
        SUB     MAX1
        BRP     NEWMAX1
        LDA     N
        STO     MAX2
loop    IN
        BRZ     END #le nombre 0 indique la fin
        STO     N
        SUB     MAX1
        BRP     NEWMAX1
        LDA     N
        SUB     MAX2
        BRP     NEWMAX2
        BR      loop

NEWMAX1 LDA     MAX1
        STO     MAX2
        LDA     N
        STO     MAX1
        BR      loop

NEWMAX2 LDA     N
        STO     MAX2
        BR      loop

END     LDA     MAX1
        OUT
        LDA     MAX2
        OUT
        HLT

MAX1    DAT     000 #Contient le plus grand
MAX2    DAT     000 #Contient le 2e plus grand
N       DAT     000 #Le nombre qu'on considère à chaque fois
```

FIGURE 3 – Programme qui lit un nombre arbitraire de nombres et renvoie les deux plus grands

Problème 6.12

Le programme LMC ci-dessous est censé additionner deux nombres, soustraire un troisième de la somme, et renvoyer le résultat, i.e. $OUT = IN_1 + IN_2 - IN_3$. *Quelle est l'erreur ? Corriger le programme.*

adresse	mnemonique	code
00	IN	901
01	STO 99	399
02	IN	901
03	ADD 99	199
04	STO 99	399
05	IN	901
06	SUB 99	299
07	OUT	902
08	HLT	000

(a) Programme avec erreur

```
00      IN
01      STO     99
02      IN
03      ADD     99
04      STO     99
05      IN
06      STO     98
07      LDA     99
08      SUB     98
09      OUT
10      HLT
```

(b) Programme corrigé

FIGURE 4 – Programme qui devrait donner : $OUT = IN_1 + IN_2 - IN_3$

En analysant le programme 4a), on peut voir que la case 99 contient bien $IN_1 + IN_2$ à la fin de la ligne 04. Par contre, par la suite, un nouveau nombre est entré **duquel on soustrait** la valeur de la case 99. Ainsi, le résultat est plutôt $OUT = IN_3 - (IN_1 + IN_2)$.

Pour régler le problème, on peut simplement garder la troisième entrée dans une autre case, *loader* la case 99 puis ensuite effectuer la soustraction avec la case où on a stocké la troisième valeur. C'est ce qui est fait aux lignes 06 à 08 du programme 4b).