Semaine 11 : architecture des ordinateurs [Solutions]

1 Un programme à étudier

a) Le résultat est le suivant :

r0	r1	r2	r3
17	123	365	473

b) Ce programme trie le contenu des registres `r0, r1, r2 et r3` pour avoir la valeur la plus petite dans `r0` et la plus grande dans `r3`. Il s'agit ici du « tri bulles » évoqué à la leçon I.1, décrit comme un des algorithmes de tri les plus simples à écrire mais aussi l'un des moins performants.

c) D'un côté, on pourrait penser que rien ne changerait : qu'on inverse ou pas deux éléments identiques ne semble pas avoir de conséquences (à part une perte de performance due à l'exécution d'une opération inutile). Le problème est que, une fois l'échange inutile fait, on met à 1 le registre `r4` pour indiquer qu'au moins un échange a été fait et qu'il faut réitérer la boucle principale ; une fois que les valeurs contenues dans les registres `r0, r1, r2 et r3` sont en ordre et si deux ou plus de ces valeurs sont identiques, on se retrouve dans une boucle infinie qui continue à échanger inutilement les valeurs identiques entre elles sans jamais terminer. Cette modification du programme serait donc une erreur !

2 Somme des valeurs absolues

Une solution possible est la suivante :

```
0: cont_ppe  0, r0, 2
1: oppose    r0         ## ou soustrait     r0, 0, r0
2: cont_ppe  0, r1, 4
3: oppose    r1
4: somme     r2, r0, r1
```

Note : « `oppose` » n'est pas difficile à imaginer : c'est le « complément à 2 » que nous avons vu en leçon I.4 !

3 Circuit

Le circuit réalise une porte XOR (« ou exclusif ») :

a	b	sortie x
0	0	0
0	1	1
1	0	1
1	1	0

Notez que le XOR correspond au bit de poids faible (« l'unité ») d'une addition de deux bits. Le bit de poids fort (la retenue) est donné par la fonction logique AND. Ainsi avec ces deux fonctions vous savez réaliser une addition de 2 bits.

4 Comparaison de valeurs horaires

Une solution possible est la suivante, si l'on pense à «`stop`» (pas vu en cours) :

```
0:      cont_pp r0, r2, 5
1:      cont_pp r2, r0, 3
2:      cont_pp r1, r3, 5
3:      charge r4, 0
4:      stop
5:      charge r4, 1
6:      stop
```

Sinon, on peut aussi faire comme cela :

```
0:      cont_pp r0, r2, 5
1:      cont_pp r2, r0, 3
2:      cont_pp r1, r3, 5
3:      charge r5, 0
4:      continue 6
5:      charge r5, 1
6:      charge r4, r5
```