Exercice 1

Expliquez les instructions suivantes :

1) `ADD R0, R1, #42`
2) `LDR R5,98`
3) `STR R0,15`
4) `CMP R4, #18`
   `BGT 77`
5) `B 100`

Exercice 2

Écrire les instructions en assembleur correspondant aux phrases suivantes :

1. Additionne la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1, le résultat est stocké dans le registre R5
2. Place la valeur stockée à l'adresse mémoire 878 dans le registre R0
3. Place le contenu du registre R0 en mémoire vive à l'adresse 124
4. Si la valeur stockée dans le registre R0 est égale 42 alors la prochaine instruction à exécuter se situe à l'adresse mémoire 85 , sinon la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 478

Exercice 3

Des entiers sont stockés aux adresses 7,8,9,10 dans la mémoire vive. Écrire une suite d'instructions en assembleur qui additionne le contenu de ces mémoires et le stocke à l'adresse 11.

Pour cela, vous n'avez le droit d'utiliser que deux registres : R0 et R1

Exercice 4

Les cases mémoires 11 12 13 de la mémoire vive contiennent des entiers. Que fait le programme suivant :

100	102	104	106	108	110	112	114	116
LDR R0,11	CMP R0,#2	BEQ 112	LDR R2,13	STR R2,30	B 116	LDR R2,12	STR R2,30	HALT

Exercice 5

Voici un programme Python, et son équivalent en assembleur.

Après avoir analysé très attentivement le programme en assembleur ci-contre, établir une correspondance entre les lignes du programme en Python et les lignes du programme en assembleur.

À quoi sert la ligne "B endif" ?

À quoi correspondent les adresses mémoires 23, 75 et 30 ?

```
x=4
y = 8
if x == 10:
    y = 9
else :
    x=x+1
z=6
```

```
   MOV R0, #4
   STR R0,30
   MOV R0, #8
   STR R0,75
   LDR R0,30
   CMP R0, #10
   BNE else
   MOV R0, #9
   STR R0,75
   B endif
else:
   LDR R0,30
   ADD R0, R0,
#1
   STR R0,30
endif:
   MOV R0, #6
   STR R0,23
   HALT
```

Assembleur (microprocesseur ARM société « *Advanced RISC Machines* »)

Rappel : un nombre (par exemple 7) est repéré avec le symbole « # » (#7)
un registre (par exemple le 7e) est repéré grâce à la lettre « R » (R7)
une adresse de la mémoire RAM (par exemple la n°7) s'écrit simplement par son numéro (7)

Les différentes instructions en assembleur :

- **LDR** R1,78 : place la valeur stockée à l'adresse mémoire 78 dans le registre R1 (toujours dans le sens RAM => registres)
- **STR** R3,125 : place la valeur stockée dans le registre R3 en mémoire vive à l'adresse 125 (toujours dans le sens registres => RAM)
- **ADD** R1,R0,#128 : additionne le nombre 128 et la valeur stockée dans le registre R0, place le résultat dans le registre R1
- **ADD** R0,R1,R2 : additionne la valeur stockée dans le registre R1 et la valeur stockée dans le registre R2, place le résultat dans le registre R0
- **SUB** R0,R1,R2 : soustrait la valeur stockée dans le registre R2 de la valeur stockée dans le registre R1, place le résultat dans le registre R0
- **MOV** R1, #23 : place le nombre 23 dans le registre R1
- **MOV** R0, R3 : déplace la valeur stockée dans le registre R3 dans le registre R0
- **B** 45 : nous avons une structure de rupture de séquence, la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 45
- **CMP** R0, #23 :Compare la valeur stockée dans le registre R0 et le nombre 23. Cette instruction CMP doit précéder une instruction de branchement conditionnel BEQ, BNE, BGT, BLT :

CMP R0, #23 **BEQ** 78 La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est **égale** à 23	**CMP** R0, #23 **BNE** 78 La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 n'est **pas égale** à 23
CMP R0, #23 **BGT** 78 La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est **plus grand** que 23	**CMP** R0, #23 **BLT** 78 La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est **plus petit** que 23

- **CMP** R0, R1 : compare la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1. Cette instruction CMP doit précéder une instruction de branchement conditionnel comme précédemment.
- **HALT** : Arrête l'exécution du programme