Langage assembleur

Exercice 1 : Pour chaque question, 4 propositions de réponses sont faites. Une seule est correcte. Laquelle ?

1.Dans quel langage les instructions sont formées de suites de 0 et de 1 ?

a) Le langage assembleur
b) Le langage machine
c) Un langage compilé
d) Un langage orienté objets

2.Laquelle des instructions Python ci-dessous a un équivalent direct en assembleur ?

a) while
b) for
c) if
d) def

3. Laquelle des instructions assembleur ci-dessous n'a pas d'équivalent en Python ?

a) ADD (additionne le contenu de deux registres)
b) MOV (Copie un nombre ou le contenu d'un registre dans un registre)
c) B (Saute à l'adresse indiquée)
d) AND (Fait un "et" bit par bit entre deux registres)

Exercice 2 – Analyses d'un programme assembleur x86

```
        section .data
tab dd 11,2,32,40,6

        section .bss
sum dd 0

        section .text
        global _start

_start:
        MOV EAX, 0
        MOV EBX, tab
        MOV ECX, 5
bcl:
        CMP ECX, 0
        JE fin
        ADD EAX, [EBX]
        ADD EBX, 4
        SUB ECX, 1
        JMP bcl
fin:
        MOV [sum], EAX
```

a. Remplir les phrases à trous

Le programme commence par allouer une zone mémoire constante contenant un tableau tab de mots de 32 *bits* initialisé avec les valeurs 11, 2, 32, 40 et 6.

Puis, dans la section .bss, une variable sum de 32 bits est initialisée à 0. C'est cette case mémoire qui va contenir la somme finale des éléments du tableau à la fin du programme.

Les instructions sont données dans la section .text et le point de départ du programme est fixé à l'étiquette _start.

Le programme commence par initialiser le registre EAX à 0. Ce registre va servir d'accumulateur pour la somme des éléments de tab.

Le registre EBX, initialisé avec l'adresse de tab, va être utilisé pour accéder successivement aux éléments du tableau.

Enfin, le registre ECX est initialisé avec la valeur 5 et va jouer le rôle de la variable de boucle.

b. Questions à choix multiples

Le corps du programme est une boucle qui s'arrête quand le compteur (le registre ECX) arrive ?

☐	à 4
☐	à 5
X	à 0
☐	à 1

Quel est le nom de l'étiquette qui représente les suites d'instruction déroulant la boucle ?

☐	_start :
X	bcl :
☐	fin :

Si la condition d'arrêt de la boucle est vérifiée, que fait le programme ?

☐	Le programme assigne la valeur 4 à la constante EBX
X	Le programme saute à la fin de la boucle représenté par l'étiquette fin

Le corps de la boucle consiste tout d'abord à ?

☐	Ajouter dans l'accumulateur EBX le contenu de la case mémoire pointée par EAX
X	Ajouter dans l'accumulateur EAX le contenu de la case mémoire pointée par EBX

c. Descriptions de la suite du programme

L'instruction :
ADD EBX, 4

Fait pointer le registre EBX sur la prochaine case du tableau. Puisque ce dernier contient des mots de 32 bits, il faut ajouter 4 à EBX pour que l'adresse corresponde au prochain mot mémoire.

Enfin, on soustrait 1 à ECX et on retourne à l'étiquette de début de la boucle bcl :

```
SUB ECX, 1
JMP bcl
```

Le programme se termine en chargeant la variable sum avec le contenu de l'accumulateur :

```
fin :
    MOV [sum], EAX
```

Exercice 3 – Que fait le programme assembleur suivant ?

```
        MOV EAX, 0
        MOV ECX, 100
ici:
        CMP ECX, 0
        JE la
        ADD EAX, ECX
        SUB ECX, 1
        JMP ici
la :
```

Corrigé :
Il calcule la somme des entiers de 0 à 100 dans le registre EAX.

Exercice 4 – Traduire les instructions suivantes en assembleur.

```
x = y + 42
if x == y :
   z = 1
else :
   z = 2
```

Pour cela, on complètera la section .text du code assembleur suivant.

```
        section .bss
x : dd 5
y : dd 10
z : dd 0
        section .text
        global _start
_start:
        MOV EAX, [y]
        ADD EAX, 42
        MOV [x], EAX
        CMP EAX, [y]
        JE then
        MOV [z], 2
        JMP fin
   then:
        MOV [z], 1
   fin:
```

Exercice 5

Voici quelques instructions de langage assembleur :

- `INP R0, 2` : entrer un nombre au clavier, et ce nombre sera stocké dans le registre 0.
- `MOV R1,R0` : Copie la valeur du registre R0 dans le registre R1.
- `ADD R2,R1,R0` : additionne les valeurs des registres R0 et R1 ; et place le résultat dans le registre R2
- `OUT R5,4` : affiche la valeur du R5 (Le 2ème paramètre 4 est la forme de sortie, 4 est la forme signée).
- `HALT` : Arrête l'exécution du programme

Voici un programme assembleur:

```
INP R3,2
MOV R4,R3
ADD R5,R4,R3
OUT R5,4
HALT
```

Dans ce programme, le résultat affiché en sortie est :

Réponses :

a) **Le double de celui entré au clavier**
b) La somme de 4 et l'entier saisi au clavier
c) La somme de l'entier saisi au clavier et d'un nombre aléatoire
d) Aucune de ces propositions