18 octobre 2022

INF1600

Travail pratique 2

Périphériques et architecture

Travail réalisé par :

El Harami, Mehdi (2113402)

Ouazani Chahdi, Rita (2178393)

Groupe Laboratoire : 02

Architecture des micro-ordinateurs (INF1600)

Département de Génie Informatique et Génie Logiciel

Polytechnique Montréal

1.2 BARÈME

TP 2		**/4,00**
Partie 2		/1,75
Q 2.2.1	/0,50	
Q 2.3.1	/0,50	
Q 2.3.2	/0,25	
Q 2.4.1	/0,50	
Partie 3		/2,25
Q 3.2.1	/1,00	
Q 3.2.2	/0,25	
Q 3.3.1	/1,00	

2.3 :

```
je printDeuxiemeLettre # print le 2eme mot considere dans cette exercice le dernier mot
divl %ecx       # premier division a 10 pour recevoir le 1 nombre
add %edx, %ebx  # on l ajoute dans le registre ebx
movl $0, %edx   # pour s assurer que edx est null
divl %ecx       # on redivise par 10 au cas ou si le asci est de 3 nombre dans le cas de lettre min
add %edx, %ebx  # on l ajoute dans le registre ebx
add %eax, %ebx  # on ajoute la quotient
movl $0, %edx   # pour s assurer que edx est null
jmp iteration   # on relance la boucle

printPremierLettre:
pushl %ebx
call printf # affiche le 1 mot
add $4,%esp
popl %ecx
movl $0, %ebx # rend la valeur de ebx = 0 pour continuer le prochain mot
```

Q2.3.1) Le programme passe par chaque caractère de "AazZWgFh tuvJKLYZHj". Pour chaque lettre nous faisons une division par 10 et on ajoute le reste de la division euclidienne dans un registre puis on fait une 2eme division par 10 sur le quotient au cas ou si le nombre ascii contient 3 chiffres. Enfin, on fait la somme des 2 divisions + le quotient. J'ai changé la valeur de string 1 a la valeur écrit dans l'exemple "ABCDX ABCDX " et j'ai bien reçu la même chose que l'exemple "BB".

2.4 :

Q2.4.1) On parle de l'opérande XOR dans ce cas-là.

On ne peut pas utiliser la clé une 2 eme fois car une fois le message est décrypté on pourrait plus utiliser le masque.

Le range de la table ASCII, que ca soit maj ou minuscule, changera donc on aura une limitation.

3 :

Q3.2.1)

```
21  addl $0x01, %eax
22  movl $0x02, %ebx
23  mull %ebx
24  subl $0x2,  %eax
25  movl %eax,  %ebx
26  movl $0x04, %eax
27  divl %ebx
28  pushl %eax
29  movl %ecx,  %eax
30  movl $0x00, %edx
31  movl $0x02, %ebx
32  divl %ebx
33  cmp $0x00,  %edx
34  jnz soustraction
35  jz addition
36
37  addition:
38  popl %eax
39  addl %eax,  valeur
40  jmp verification
41
42  soustraction:
43  popl %eax
44  subl %eax,  valeur
45
46  verification:
```

En utilisant Euclere on peut arriver a approximer la valeur de pi grâce a la formule fournit dans l'énonce (formule de Leibniz). Ce qu'on a fait dans notre code c'est d'additionner puis soustracter puis refaire cela en boucle jusqu'à finir toutes les itérations demandés.