مجلس الخدمة المدنية
إدارة الموظفيـــــن
اللـجنة الفاحصة

مباراة لملء بعض الوظائف الشاغرة في ملاك مؤسسة كهرباء لبنان

لوظيفة: رئيس قسم ممتاز أو مراقب معاون أو أمين صندوق مركزي أو أمين مخزن مركزي (اختصاص معلوماتية)

مسابقة في: أنظمة تشغيل الكمبيوتر Operating Systems **المدة:** ساعتان

Exercice 1.

Consider a main memory system that consists of a number of memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the date.
The internal operation of different memory modules may overlap in time, but only one request can be on the bus at any time.
Sketch a graph of the maximum write rate (words per second) as a function of the module cycle time, assuming eight memory modules and a fixed bus busy time of 100 ns.

Exercice 2.

a. Consider the following C/Unix program::

```
void main ()
{       int pid;
        pid = fork ();
        printf ("%d \n", pid);
}
```

What are the possible outputs, assuming the *fork()* succeeds?

b. What happens when executing the following program?

```
void main ()
{
        for ( ; ; )
                fork () ;
}
```

Exercise 1.

Considérons un système de mémoire principale qui se compose d'un certain nombre de modules de mémoire attachés au bus système qui est de largeur un mot. Quand une demande d'écriture est faite, le bus est occupé pendant 100 nanosecondes (ns) par les signaux de données, d'adresse et de commande. Au cours de ces 100 ns, et pour 500 ns supplémentaires, le module de mémoire adressé exécute un cycle recevant et stockant les données.
Le fonctionnement interne des différents modules de mémoire peut subir des chevauchements dans le temps, mais une seule demande peut être sur le bus à un moment donné
Dessiner un graphe de la vitesse maximale d'écriture (mots par seconde) comme une fonction du temps de cycle du module, en supposant qu'il y a huit modules de mémoire et un temps fixe d'occupation du bus de 100 ns.

Exercise 2.

a. Considérons le programme C/Unix suivant:

```
void main ()
{       int pid;
        pid = fork ();
        printf ("%d \n", pid);
}
```

Quels sont les résultats possibles, en supposant que l'appel *fork()* réussit?

b. Qu'arriverait-il lorsqu'on exécute le programme suivant?

```
void main ()
{
        for ( ; ; )
                fork () ;
}
```

Exercice 3.

Consider the following state (snapshot) of a system. There are no outstanding (unsatisfied) requests for resources.

available			
r1	r2	r3	r4
1	5	2	0

	current allocation				maximum demand			
process	r1	r2	r3	r4	r1	r2	r3	r4
p0	0	0	1	2	0	0	1	2
p1	1	0	0	1	1	5	1	1
p2	1	3	0	4	2	3	5	6
p3	0	6	3	2	0	6	7	2
p4	0	0	1	4	0	6	5	6

Is the system safe? Explain.

Exercice 4.

Suppose the following two processes, ***"foo"*** and ***"bar"*** are executed concurrently and that they share the semaphore variables S and R (each initialized to 1) and the integer variable x (initialized to 0).

```
void foo( ) {
    do {
        semWait(S);
        semWait(R);
        x++;

    semSignal(S);

    SemSignal(R);
    } while (1);
}
```

```
void bar( ) {
do {
    semWait(R);
    semWait(S);
    x--;
    semSignal(S;
    SemSignal(R);
} while (1);
}
```

a. Give an execution sequence of programs statements in which one or both processes are blocked forever.
b. Could the concurrent execution of these two processes result in the indefinite postponement of one of them? if yes, give an execution sequence in which one is indefinitely postponed.

Exercise 3.

Considérons l'état suivant (snapshot) d'un système. Il n'y a pas de demandes en suspens (insatisfaits) en matière de ressources

available			
r1	r2	r3	r4
1	5	2	0

	current allocation				maximum demand			
process	r1	r2	r3	r4	r1	r2	r3	r4
p0	0	0	1	2	0	0	1	2
p1	1	0	0	1	1	5	1	1
p2	1	3	0	4	2	3	5	6
p3	0	6	3	2	0	6	7	2
p4	0	0	1	4	0	6	5	6

Le système est-il sûr? Expliquer.

Exercise 4.

Supposons que les deux processus suivants, "***foo"*** et ***"bar"*** soient exécutés en concurrence et qu'ils partagent les sémaphores S et R (chacun initialisé à 1) et la variable entière x (initialisée à 0).

```
void foo( ) {
    do {
        semWait(S);
        semWait(R);
        x++;

    semSignal(S);

    SemSignal(R);
    } while (1);
}
```

```
void bar( ) {
do {
    semWait(R);
    semWait(S);
    x--;
    semSignal(S;
    SemSignal(R);
} while (1);
}
```

a. Donner une séquence d'exécution des instructions des programmes dans laquelle un ou les deux processus sont bloqués pour toujours.
b. Est-ce que l'exécution concurrente de ces deux processus peut avoir comme résultat une suspension indéfinie de l'un des d'eux ? Si oui, donnez une séquence d'exécution dans laquelle un des deux est suspendu indéfiniment.

Exercice 5.

Consider the following pseudo assembly code for computing c = a + b. Assume that a, b, and c are assigned to consecutive memory "words" (memory is generally addressed byte by byte and assume that a word is 4 bytes wide) and address for "a" is 0x0000EC00. Also, we initially have a = 22, b = 158, and c = 0. Assume that the first instruction of the code is stored in 0x0000B128. Also, each instruction has the OpCode in the first byte (most significant byte) and the remaining 3 bytes specify the corresponding address. The OpCode for storing a value is 1, load is 2, and add is 3.

```
0x0000b128 load a
0x0000b12c add b
0x0000b130 store c
```

Show the memory addresses and contents for all the instructions and involved data. Use the format as follows to express your answer (but the following is not the answer). For all data, use hexadecimal representation.

addresses	**contents**
0x00002104	0x00000001
0x00002108	0x00000002

Exercise 5.

Considérons le pseudo code assembleur suivant pour le calcul de c = a + b. Supposons que a, b et c sont affectées à des "mots" consécutifs d'une mémoire séquentielle (la mémoire est généralement adressée octet par octet et le mot est d longueur 4 octets) et l'adresse de "a" est 0x0000EC00. De plus, initialement, a = 22, b =158, et c = 0. Supposons que la première instruction du code est stockée dans 0x0000B128. En outre, chaque instruction a l' OpCode dans le premier octet (octet le plus significatif) et les 3 octets restants spécifient l'adresse correspondante. L'OpCode pour stocker une valeur est 1 (store), pour le chargement est 2 (load), et pour l'addition est 3 (add).

```
0x0000b128 load a
0x0000b12c add b
0x0000b130 store c
```

Présenter les adresses mémoire et les contenus pour toutes les instructions et les données impliqués. Utilisez le format suivant pour exprimer votre réponse (mais ce qui suit n'est pas la réponse). Pour toutes les données, utiliser la représentation hexadécimale.

adresses	**contenus**
0x00002104	0x00000001
0x00002108	0x00000002

بيروت في 2013/1/19 اللجنة الفاحصة