GRO 300 - Examen formatif

Processus, fils, et parallélisation du travail (GRO 300-1)

Question 1

Quelle est la relation entre processus et fils ?

Question 2

Est-ce qu'il existe une limite au nombre de fils pouvant être exécutés simultanément ? Si oui, laquelle ? Sinon, pourquoi ? Détaillez votre raisonnement.

Question 3

Nommez un inconvénient de la communication inter-processus par rapport à la communication inter-fils.

Question 4

Est-ce qu'un fil d'exécution peut avoir un espace mémoire bien à lui ? Donnez un exemple en C++.

Question 5

Qu'est-ce qui peut compliquer la parallélisation du traitement de plusieurs sous-tâches ?

Question 6

Vous êtes ingénieur dans une compagnie qui souhaite utiliser un robot manipulateur pour détecter des pièces défaillantes sur un convoyeur. Un de vos collègues a déjà développé un prototype du système de traitement qui se divise en quatre parties séquentielles :

1. Le rehaussement du contraste de l'image et la suppression de la couleur du convoyeur, pour qu'il ne reste que les objets à évaluer dans l'image. Le traitement est isolé par pixel, c.-à-d. que, pour chaque pixel, le traitement ne dépend pas des pixels voisins ;
2. La détection de points d'intérêts distinctifs dans toutes l'image, comme des coins ou des traits. Cette étape consiste à analyser chaque pixel avec ses voisins de l'image d'origine dans un rayon de 15 pixels. L'analyse par pixel est séquentielle ;
3. Le regroupement des points d'intérêts en objets, en supposant que les points d'intérêts près les uns des autres appartiennent au même objet. La tâche est itérative : Pour *N* points d'intérêts, on connait d'abord le nombre d'objets à détecter (*P*, par exemple 3). On sélectionne *P* points au hasard dans l'image qu'on considère être les centres de ces objets. Ensuite, pour chaque point *n* dans l'ensemble *N*, on trouve le point *p* de l'ensemble *P* qui lui est le plus proche, et on associe ce point à l'objet *p*. Finalement, pour chaque objet *p,* on calcule le barycentre des points (c.-à-d. la moyenne des coordonnées, ce qui devrait correspondre au centre de l'objet) qu'ils lui sont associés, ce qui devient *p',* le nouveau centre de l'objet *p*. On répète le processus d'association des points *N* au objets *P* jusqu'à ce que les ensembles ne changent plus.
4. L'analyse statistique de l'appartenance des points d'intérêts aux objets. Les objets ne possédant pas les bons signes distinctifs sont considérés comme étant défaillants.

Vous avez la responsabilité d'optimiser le prototype en place et d'augmenter le débit du système de traitement de données, l'objectif étant de détecter le plus d'objets défaillants à la minute. On vous suggère d'utiliser la programmation concurrente pour mieux exploiter les capacités de l'ordinateur embarqué de la chaîne de montage.

Répondez aux questions suivantes comme si le système n'avait qu'à traiter une seule image :

a) Peut-on faire une séparation des données en parallèle dans les parties 1 et 2 ? Si oui, comment ? Sinon, pourquoi ?

b) Suggérez au moins deux façons de paralléliser le traitement de l'étape 3.

c) Croyez-vous qu'il soit possible d'améliorer la performance de l'algorithme **sur une seule image** en parallélisant les quatre parties entre elles ? Expliquez pourquoi et comment.

Mécanismes de synchronisation (GRO-300-1)

Question 7

En C++, pourquoi préconise-t-on l'usage de verrous comme `std::lock_guard` et `std::unique_lock` plutôt que d'appeler les fonctions « `lock()` » et « `unlock()` » directement sur un mutex ?

Question 8

Soit cet exemple (chaque colonne s'exécute dans un fil différent) :

```
int compteur_ = 0;  // Variable globale accessible aux deux fils.
```

```
void fil_A()
{
  while (true) {
    compteur_++;          // A1

    // Pause 1000 ms :
    usleep(1000 * 1000);
  }
}
```

```
void fil_B()
{
  while (true) {
    compteur_++;                  // B1
    printf("Nb. d'accès: %d\n", // B2
           compteur_);
    usleep(1000 * 1000);
  }
}
```

Est-ce que la ligne de code pointée par "A1" représente une condition critique ? Si oui, comment peut-on éliminer cette condition ?

Question 9

Une fois toutes les sections critiques protégées dans ce code, est-ce la sortie du programme (l'affichage en ligne B2) sera toujours régulière (c.-à-d. des nombres entiers augmentant par 2 à chaque fois) ? Pourquoi ?

Question 10

À quel moment il est plus intéressé d'utiliser un système de variable de condition au lieux d'utiliser seulement des verrous ?

Mesures de performance (GRO-300-2)

Question 11

Nommez au moins deux raisons pour lesquelles le temps d'exécution d'un programme sur l'unité centrale (CPU) de votre système peut différer du temps réellement écoulé pour réaliser une tâche.

Question 12

Pour une unité de traitement donnée, comment peut-on mesurer de façon absolue la durée d'exécution d'un programme ?

Question 13

Un programme a été décomposé en trois classes d'instructions (A, B, et C) en donnant la proportion d'instructions appartenant à chaque classe. Le nombre de cycles d'horloge par instruction est donné pour deux unités centrales différentes (1 et 2).

Classe d'instruction	Proportion du programme	Cycles sur unité 1	Cycles sur unité 2
A	10 %	2	3
B	30 %	3	4
C	60 %	2	3

Si les unités 1 et 2 sont cadencées à 3,2 GHz et 3,1 GHz respectivement, laquelle exécutera le programme plus rapidement ? Détaillez vos calculs.

Pipelines (GRO-300-2)

Question 14

Est-ce qu'un pipeline augmente le débit ou la vitesse d'exécution des instructions d'un système ?

Question 15

Donnez un exemple d'aléa de données dans un pipeline (*data hazard*).

Question 16

Qu'est-ce qui distingue un aléa de données d'un aléa de contrôle ?

Mémoire (GRO-300-2)

Question 17

Soit ce programme en C++:

```
00 const int LEN_S = 65536;
01 double signal_org[LEN_S] = {…}; // signal_org contient des
mesures.
02 double signal_fin[LEN_S];
03 double noyau[3] = {0.25, 0.50, 0.25};
04
05 signal_fin[0] = signal_org[0];
06 for (int i = 1; i < (LEN_S - 1); ++i) {
07   signal_fin[i] = 0.0;
08   signal_fin[i] += signal_org[i - 1] * noyau[0];
09   signal_fin[i] += signal_org[i] * noyau[1];
10   signal_fin[i] += signal_org[i + 1] * noyau[2];
11 }
12 signal_fin[LEN_S-1] = signal_org[LEN_S - 1];
```

Supposons que les accès en mémoire de ce programme compilé sont optimisés selon le principe de localité, mais seulement à partir de la ligne 5. Indiquez lesquelles des localités (aucune, spatiale, temporelle, ou les deux) ont été considérées et pourquoi, pour :

a) `signal_org[i - 1]` à la première exécution de la ligne 8 (`i = 1`) ?

b) `signal_org[i]` à la première exécution de la ligne 9 (`i = 1`) ?

c) `noyau[1]` à la première exécution de la ligne 9 (`i = 1`) ?

d) noyau[1] à la seconde exécution de la ligne 9 (`i = 2`) ?

Question 18

En IEEE 754, comment est représenté 20,1 ?

Instructions, registres, et cycles d'horloge (GRO-300-2)

Question 19

Soit ce programme en C++:

```
float prix_total;
float code_de_article = 10; // Fruits et Légumes
float prix = 42.69;
float tps = 0.05;
float tvq = 0.09975;

if (code_de_article != 10) {
    float prix_avec_tps = prix * tps;
    float prix_avec_tvq = prix * tvq;
    prix_total = prix_avec_tps + prix_avec_tvq;
} else {
    prix_total = prix;
}
```

Vous disposez de ces instructions, de 6 registres (R1 à R6), et d'une mémoire accessible par nom de variable :

- LDC Rx, VAL # Charge la constante VAL dans le registre Rx
- STO $A, Rx # Stocke Rx en mémoire à la variable $A
- ADD Ra, Rb # Ra = Ra + Rb
- SUB Ra, Rb # Ra = Ra – Rb
- MUL Ra, Rb # Ra = Ra * Rb
- BNE N, Ra, VAL # Branchement à l'adresse N si Ra != VAL
- BE N, Ra, VAL # Branchement à l'adresse N si Ra == VAL

Traduisez le programme en C en pseudo-assembleur à l'aide des instructions données. Associez d'abord chaque variable à un registre séparé en enregistre.