TD2, M1 Info, Parallélisme, 2009-2010

Exercice 1

Ecrire un algorithme qui calcule l'expression

y := 2* ((a+b) / (c - d) + e*f) + (a+b) * (c-d)

en évaluant en parallèle des sous-expressions contenant une opération et en utilisant les pseudo-instructions "parbegin" et "parend" (on pourra utiliser un nombre quelconque de variables intermédiaires). Constuire un système équivalent de parallélisme maximal.

Exercice 2 :

La notion de graphe de dépendance permet de décrire les types de dépendances entre données et d'extraire le meilleur parallélisme possible d'un programme.
Il existe une condition suffisante d'indépendance entre différentes instructions ou parties de code. Cette condition est connue sous le nom de condition de Bernstein et s'appuie sur les notions de variables lues et de variables modifiées.
Soit S une séquence d'instructions. On définit L(S) comme l'ensemble des variables utilisées en lecture dans le code S, et M(S) l'ensemble des variables qui subissent une affectation dans le code S.
La **condition de Bernstein** s'énonce ainsi. Deux instructions (ou plus généralement, 2 séquences de code symbolisées par 2 taches) peuvent être exécutées dans un ordre quelconque, ou en parallèle (cad qu'elles n'interfèrent pas), sans modifier le résultat obtenu dès lors que l'on a les trois conditions suivantes.

$M(S1) \cap L(S2) = Ø$ appelée dépendance vraie (S2 ne doit pas du tout utiliser de valeur produite par S1 tant que S1 ne s'est pas terminée)

$L(S1) \cap M(S2) = Ø$ appelée anti-dépendance (S2 ne doit pas modifier trop vite ses variables si S1 doit d'abord lire leur ancienne valeur)

$M(S1) \cap M(S2) = Ø$ appelée dépendance de sortie (résultat indéterminé si S1 et S2 s'exécutent en parallèle)

Si tel n'est pas le cas, les 2 taches sont dépendantes l'une de l'autre, et ceci se matérialise par un arc entre elles dans le graphe de taches.

Question 2.1

En vous servant de la définition de la condition de Bernstein, montrer que la parallélisation de la boucle « pour » (une tâche par tour de boucle) est impossible. Identifier précisément le nom de la dépendance qui est mise en jeu dans votre argumentation. Que calcule ce bout de code ?

```
a[0]=0
sp[0]=a[0]
pour i=1 jusqu'à n-1 faire
        S1 : a[i]=2*i
        S2 : sp[i]=a[i] + sp[i-1]
Finpour
```

Même si la condition de Bernstein n'est pas satisfaite sur cet exemple, cela ne signifie pas pour autant que le code ne puisse être exécuté en parallèle (car la condition est suffisante mais

non nécessaire). Sans modifier fondamentalement le code ci-dessus, on ne peut cependant rien faire. Par contre, nous pourrions essayer de réorganiser le corps de la boucle afin d'extraire la séquence d'instructions qui pourrait elle être parallélisée. (cherchez laquelle est-ce) : c'est typiquement ce qu'un compilateur basé sur de la parallélisation automatique pourrait réussir à faire.

Question 2.2
Refaire la question 2.1 avec le bout de code suivant (qui ne calcule pas la même chose) :

```
pour i=1 jusqu'à n faire
        S1 : a[i]=2*i
        S2:  sp[i]=a[i]
finpour
pour i=1 jusqu'à n-1 faire
        S3: sp[i]=a[i] + sp[i+1]
Finpour
```

Question 2.3
La condition de Bernstein nous permet d'avoir la définition ci-dessous :
Un graphe de taches est de parallélisme maximal si il vérifie la propriété suivante : la suppression de tout arc (T,T') du graphe entraine la non-indépendance, autrement dit l'interférence des taches T et T'. Montrer de manière intuitive que le graphe de taches obtenu à l'exo 1 est bien de parallélisme maximal.

Exercice 3 *(à rendre par email, individuellement, avant le cours/TD suivant)*:

Donner le graphe de dépendance qui correspond au programme suivant (sachant que l'enchainement en séquence introduit une dépendance éventuellement « fausse » entre les taches respectives des 2 séquences) :

```
debut
        parbegin lire(a) ; lire(b) ; parend
        parbegin c=a*a; debut d=b*b;
                                parbegin f=a+b; g=a*b; parend
                        fin
        parend
        e:=c+d-(g+f)
fin
```

Le graphe est-il de parallélisme maximal ?

Exercice 4

Question 4.1
Ecrire un programme parallèle faisant intervenir 3 processus pour multiplier entre elles deux matrices de taille 3x3, notées A, B, dont le résultat est mis dans une matrice C.

Question 4.2
Faire de même en utilisant le maximum de processus afin d'avoir la plus grande quantité de taches parallèles possibles mises en œuvre. Quelle est la quantité totale de processeurs (virtuels) nécessaires ? Quel est l'ordre de grandeur de la durée d'exécution ?