TD 7 Parallélisme de contrôle : espace d'adressage unique

Dans tous les exercices suivants, on considère une architecture multiprocesseur à mémoire partagée, espace d'adressage unique, organisée sur un bus unique, et assurant la cohérence de cache.

7.1 Consistance mémoire

La fig. 7.1 décrit les accès mémoires concurrents de deux processeurs. `A`, `F` et `G` sont des variables, `R1` et `R2` des registres. Initialement, `A`, `F` et `G` sont à 0.

p_0	p_1
`F = 1`	`G = 1`
`A = 1`	`A = 2`
`R1 = A`	`R3 = A`
`R2 = G`	`R4 = F`

FIG. 1 – Test de consistance.

1. Quels sont les résultats possibles sur `R1`,`R2`,`R3`,`R4` ?
2. On considère une mémoire séquentiellement consistante. Montrer que le résultat ne peut pas être `R2` = `R4` = 0. Quelle caractéristique architecturale des microprocesseurs actuels pourrait produire ce scénario ?
3. On considère une mémoire séquentiellement consistante. Quels sont les résultats sur `R2` et `R4` compatibles avec `R1` = 1 et `R3` = 2 ?

7.2 Protocole MESI

Soit un multiprocesseur à 3 processeurs. Les caches sont initialement vides. Un succès cache lecture ou écriture prend 1 cycle. Les transactions simples sur le bus prennent 60 cycles et les transactions nécessitant le transfert d'une ligne prennent 90 cycles. Quel est le coût de la séquence

$$r_1 w_1 r_1 w_1 r_2 w_2 r_2 w_2 r_3 w_3 r_3 w_3$$

où les r_i (respectivement w_i) sont la lecture (respectivement l'écriture) par le proceseur p_i d'une adresse mémoire unique.

7.3 Verrous

L'attente active sur un verrou est réalisé par le code suivant, de type *TestTestAndSet* :

```
deb : ll      r1, lck          /* Chargement lie */
      bnz     deb              /* Verrou deja occupe */
      add     r2, r2, 1        /* Valeur occupe */
      sc      r2, lck          /* Rangement conditionnel, retour dans r2 */
      bne     r2, deb          /* Echec rangement */
```

Tous les processeurs doivent exécuter une section critique contrôlée par le verrou `lck`. On veut évaluer le coût de synchronisation de cette situation.

Initialement, le processeur p_0 possède le verrou, et tous les processeurs possèdent un copie de la variable `lck` dans leur cache.

Le coût de toute transaction bus est 50 cycles processeur. Les transactions bus sont atomiques et séquentielles (ni pipelinées, ni entrelacées). Le bus est parfaitement équitable : les requêtes sont servies dans l'ordre d'apparition. Les processeurs fonctionnent à la même vitesse.

1. Pour 3 processeurs, décrire les étapes de la mise à jour du verrou lors de sa libération par p_0 (figure 1).

Etape	p_0	Etat	p_1	Etat	p_2	Etat	Activite bus
1	A le verrou	S	lit 1	S	lit 1	S	rien
2	Ecrit 0						
...							

FIG. 2 – Trafic de cohérence à la libération d'un verrou. Les colonnes *Etat* décrivent l'état MESI de la ligne de cache correspondant au verrou

2. Quelle est le nombre de transactions bus et le coût en cycles à 20 processeurs ?

7.4 Réduction

L'algorithme séquentiel ci-dessous calcule la somme d'un tableau de $N = 2^n$ nombres.

```
M = N/2
do k = 1, n
   do i = 0, M-1
      a(i) = a(i) + a(i + M)
   end do
M = M/2
end do
```

1. Ecrire le code SPMD parallèle de l'algorithme sur $P = 4$ processeurs.
2. A quelle étape de cet algorithme le problème du faux partage intervient-il ?