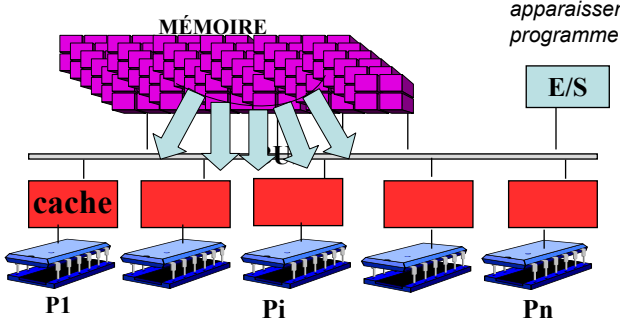
Consistance séquentielle

- Une lecture d'un mot mémoire doit retourner la valeur de la dernière écriture dans ce mot, et ceci pour tous les processeurs

Toutes les opérations mémoires apparaissent s'effectuer atomiquement et les opérations émises par un seul proc apparaissent s'effectuer dans l'ordre du programme



Impact

- Interdit de nombreuses opérations d'optimisation réalisées par les compilateurs et les procs modernes
- Le modèle de consistance mémoire impacte :
  - Le programmeur (validité des opérations de synchro entre flots)
  - La performance (optimisation logicielle et matérielle)
  - La portabilité (selon les modèles utilisés par les # procs)
- Nécessaire à chaque niveau d'interface entre le programmeur et le système
  - Processeur (hard + langage machine)
  - Langage évolué (primitives de synchro et transformation en instructions machine)

Sémantique pour un monoprocesseur

- Pour un programme séquentiel il suffit de maintenir les dépendances de données et de contrôle pour assurer la définition de la cohérence séquentielle.
  - En conséquence le compilateur et le matériel peuvent exécuter une liste d'instructions dans un ordre différent de l'ordre initial.
  - Les accès mémoires sont différés, recouverts ou réordonnés pour améliorer les performances.

Définition pour un multiprocesseur

- Lamport 1979

« un système multiprocesseur assure la consistance séquentielle si le résultat de l'exécution de multiple flots est identique à une exécution séquentielle de l'ensemble des opérations sur la mémoire et les opérations émises par chaque processeur s'exécutent dans l'ordre prévu »

- Conséquence
  - Maintenir l'ordre des dépendances du programme séquentiel
  - Maintenir un ordre sur l'ensemble des écritures émises par tous les processeurs (écritures atomiques)

Exemples

- L'algorithme de Dekker pour l'entrée en SC

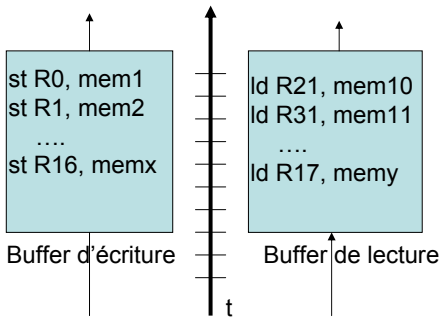
```
P1                              P2
Flag1 = 1                       Flag2 = 1
If (flag2 == 0)                 If (flag1 == 0)
        entrée en SC                    entrée en SC
```

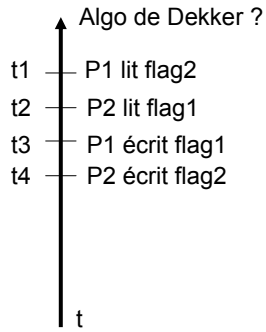
- ex2

```
P1          P2              P3
A = 1
            If (A ==1)
                B=1
                            If (B ==1)
                                register1=A
```


Architectures sans cache

• La consommation des deux files s'effectue sur un seul proc en respectant seulement les règles de dépendances sur des variables communes.





Conséquence pour P1 et P2 ?

• Opération de lecture non bloquante (spéculation)
• Architecture multi bus permettant une simultanéité d'écriture

Architectures avec caches

- Une lecture après une écriture peut être servie avant la fin de l'écriture.
- La réplication de données partagées introduit trois problèmes complémentaires :
  - Le protocole de cohérence de cache doit propager une écriture à tous les caches qui contiennent une copie de l'écriture
  - La détection d'une fin d'écriture (pour préserver l'ordre d'exécution des instructions) engendre de nouvelles transactions.
  - La propagation d'écriture sur de multiples copies est non atomique
- Un modèle de cohérence en présence de cache définit un intervalle de temps pour propager une écriture vers tous les processeurs.

- Détecter la fin des écritures
  - Sur une écriture sur une ligne de cache partagée entre plusieurs processeurs, le système nécessite un mécanisme d'acquittement des messages d'invalidation ou de mise à jour des caches cibles. De plus, les messages d'acquittement doivent être récupérés (au niveau de la mémoire ou du proc qui émet l'écriture).
  - Le proc qui émet la demande doit être notifié de la fin de celle-ci.
  - Un proc ne peut considérer que l'écriture est terminée qu'après avoir reçu cette notification.
- Maintenir l'illusion de l'atomicité des écritures
  - La propagation vers tous les caches est non-atomique.
  - Imposer que les écritures pour le même mot mémoire soient sérialisées
  - Interdire une lecture d'un mot venant d'être écrit tant que tous les caches qui contiennent ce mot n'ont pas acquitté le message d'invalidation ou de m.a.j généré par l'écriture.