TD 11 4MMARCA, Ensimag 2A/S6, ISI/SEOC

Opérations atomiques

Architecture avancée, 2018-2019

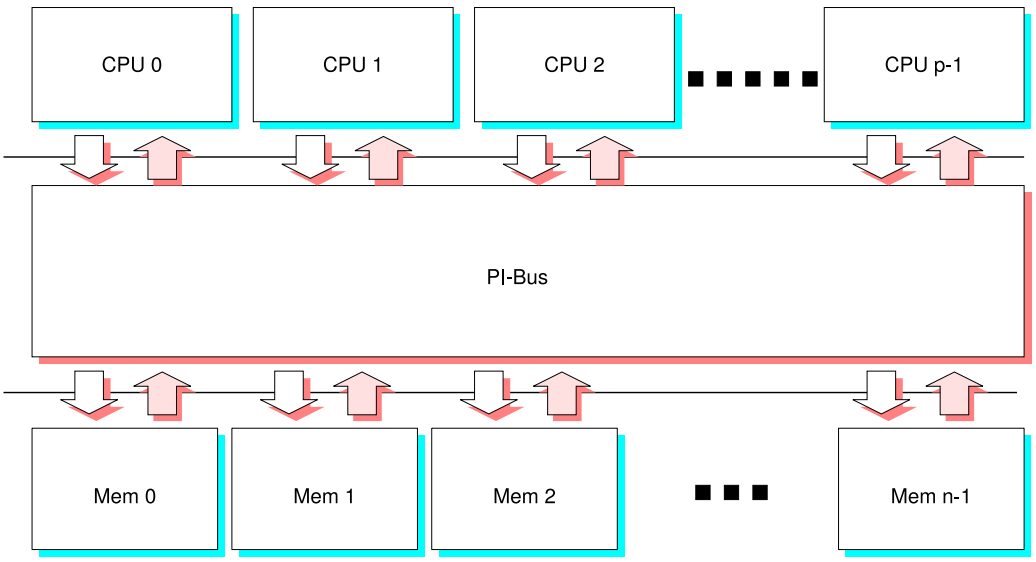
Ex. 1 : Test and set

Le *test-and-set* est l'opération atomique la plus simple, qui est en fait une lecture suivie d'une écriture modifiante (*read-modify-write* en anglais). Son principe est de lire une variable et de la mettre à **1** sans qu'il soit possible à un autre maître du système de faire un accès entre ces deux actions. Ainsi, si la valeur retournée est **0**, le programme considère qu'il a acquis la variable (qu'on appelle alors verrou). Si au contraire la valeur retournée est **1**, alors c'est qu'un autre programme possède le verrou.

Question 1 – Soit la fonction C ayant le prototype suivant `bool test_and_set(bool *lock)`. Ecrivez en C le code de la fonction `void critical_section(void)` permettant d'exécuter une section critique :

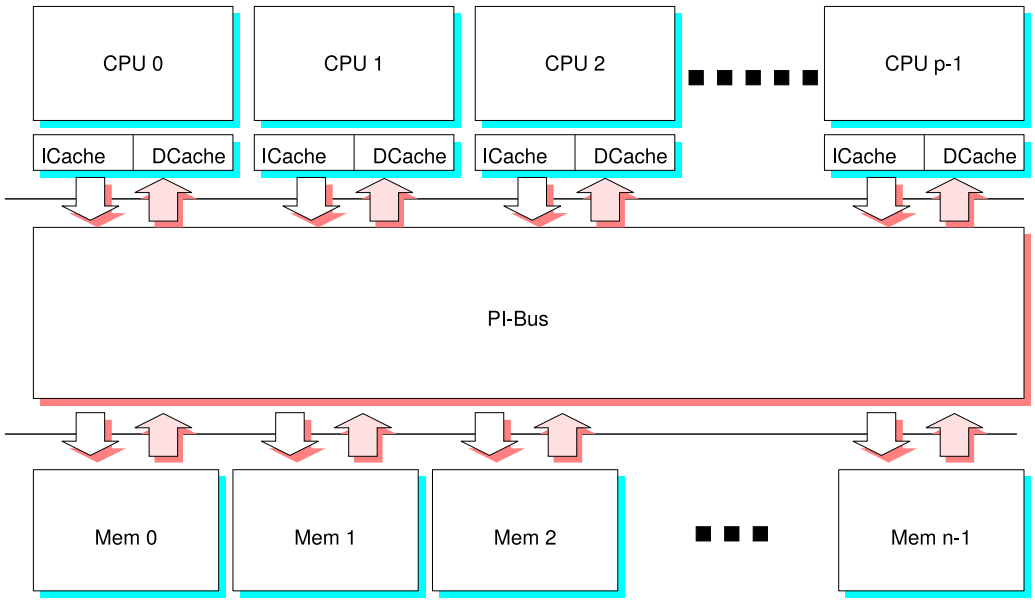
```
volatile bool lock;
void critical_section(void)
{
   /* Prise du verrou: à écrire */
   code de la section critique exécuté par un unique programme
   /* Relâchement du verrou: à écrire */
}
```

On considère un système construit autour d'un bus PI (*c.f.* le TD 1) doté de p processeurs et n mémoires. Dans un premier temps, on fera l'hypothèse que ces processeurs ne possèdent pas de caches, comme illustré ci-dessous.



Question 2 – En supposant que le processeur ait une requête `RW=101` indiquant qu'une instruction `test_and_set` est requise par le processeur, proposez une machine d'état qui réalise l'instruction de manière atomique. On se contentera de donner l'automate sans préciser la partie opérative, et l'on décrira donc textuellement les opérations réalisées dans les états.

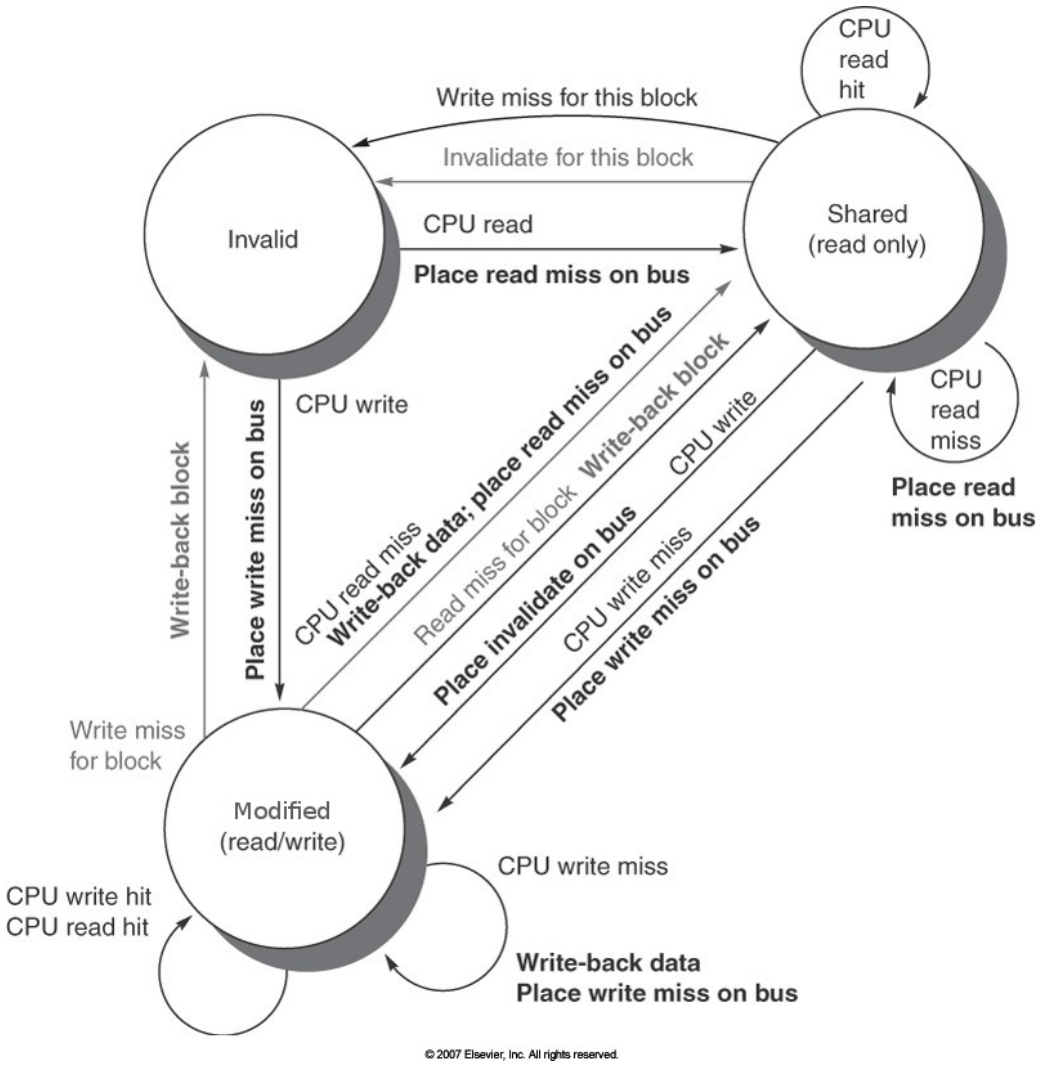
On ajoute des caches aux processeurs, comme indiqué ci-dessous.



Question 3 – Précisez ce qui change à cause des caches. Mettez en évidence le dysfonctionnement avec un exemple.

Les deux solutions envisageables pour éviter ce problème sont a) ne pas passer par le cache pour les instructions `test_and_set`, et b) modifier le protocole de cohérence de cache.
Question 4 – Donnez les avantages et inconvénients des deux solutions.

On se propose de modifier le protocole de cohérence de cache.



Question 5 – Quelles modifications faut-il apporter au protocole de cohérence MSI pour avoir un `test_and_set` fonctionnel ?

Ex. 2 : LLSC

Les instructions `ll` et `sc` permettent également des accès atomiques à des données, mais avec un mode opératoire différent et théoriquement plus efficace. Ces deux instructions fonctionnent par paires. Le `ll` (load link) retourne la valeur courante d'un emplacement mémoire. Le `sc` (store conditional) associé écrit une nouvelle valeur dans ce même emplacement si aucune modification n'y est intervenue depuis le `ll`. Le registre de donnée du `sc` indique le succès de la paire d'instruction après son exécution (**0** en cas d'échec, **1** en cas de succès).

Question 1 – Quelles sont les conditions pouvant faire échouer un `sc` ?

On fait l'hypothèse que l'on a pas de cache.

Question 2 – Précisez ce qui doit être fait du coté du processeur qui fait le `ll` et le matériel que cela induit.

On ajoute des caches aux processeurs, comme indiqué ci-dessous.

Question 3 – Traduisez les en actions relative au protocole de cohérence.

Question 4 – Modifiez l'automate de cohérence pour les prendre en compte.