Modélisation et vérification de systèmes concurrents

Architecture multiprocesseur à mémoire partagée

Kimmeng Ly Max Eliet

Sorbonne Université Sciences

Encadrante : E. Encrenaz

Introduction

On considère un système multiprocesseur à mémoire partagée. Le système est muni d'une hiérarchie mémoire à deux niveaux :

Introduction

On considère un système multiprocesseur à mémoire partagée. Le système est muni d'une hiérarchie mémoire à deux niveaux :

Mémoire centrale

Stocke les instructions, données, pile des programmes en cours d'exécution et du système d'exploitation.

Introduction

On considère un système multiprocesseur à mémoire partagée. Le système est muni d'une hiérarchie mémoire à deux niveaux :

Mémoire centrale

Stocke les instructions, données, pile des programmes en cours d'exécution et du système d'exploitation.

Caches privés

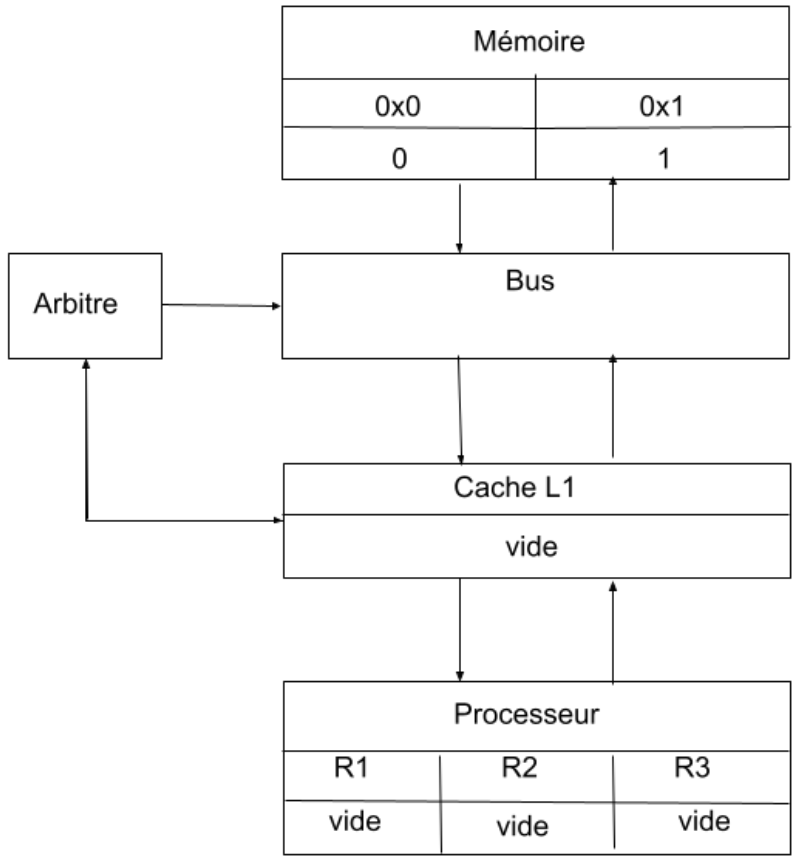
Associés à chaque processeur et comprennent deux parties distinctes :

- Caches d'instructions
- Caches de données

Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

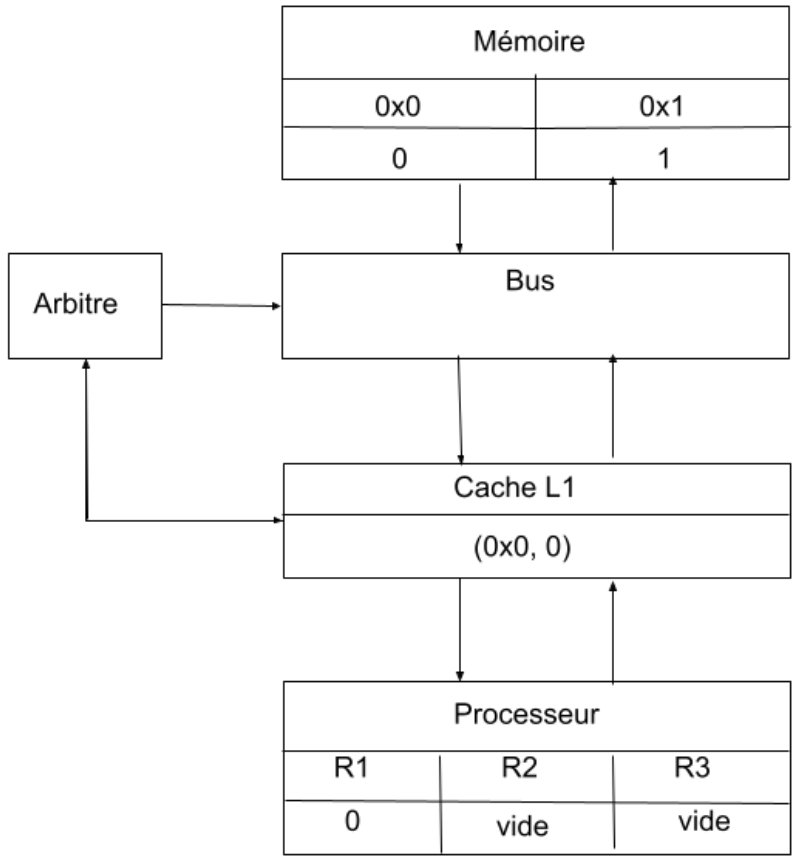


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

- ld R1, [0]

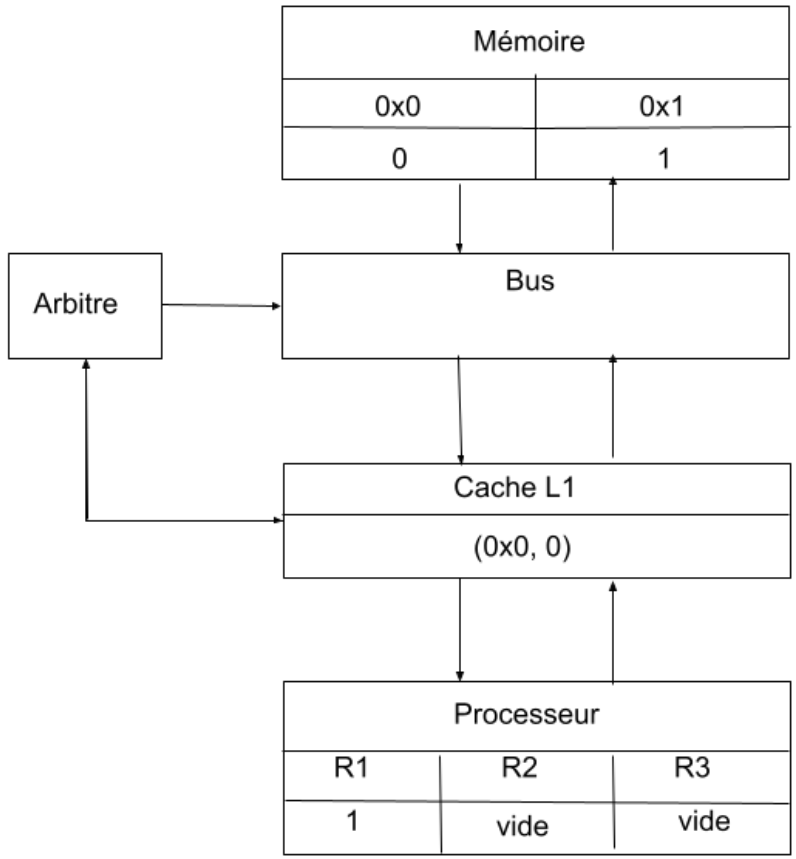


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

- ld R1, [0]
- add R1, R1, 1

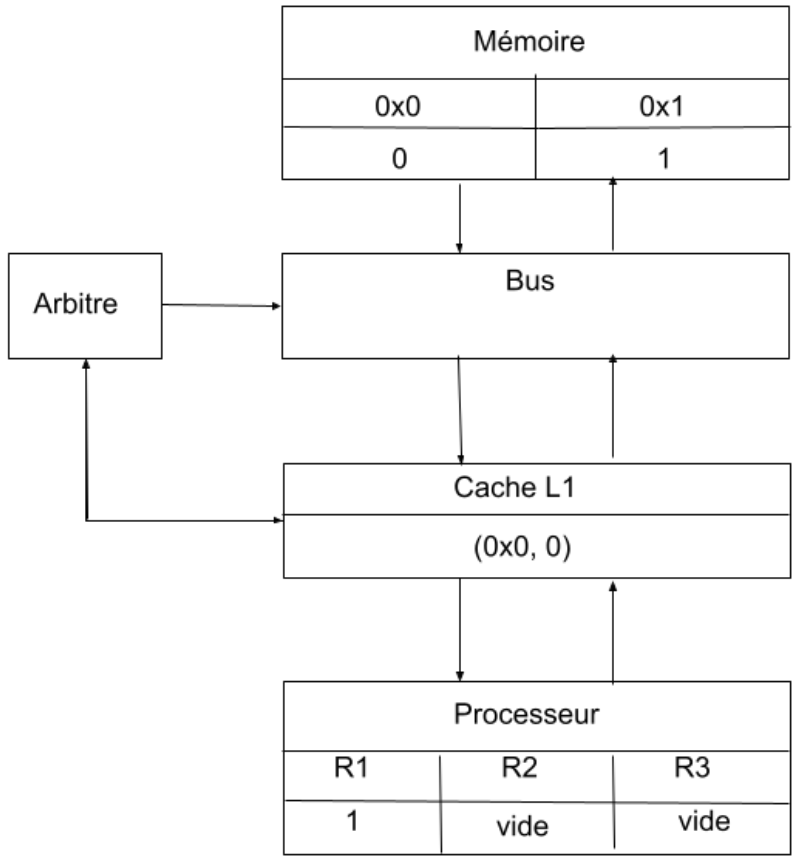


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

- ld R1, [0]
- add R1, R1, 1
- st R1, [1]

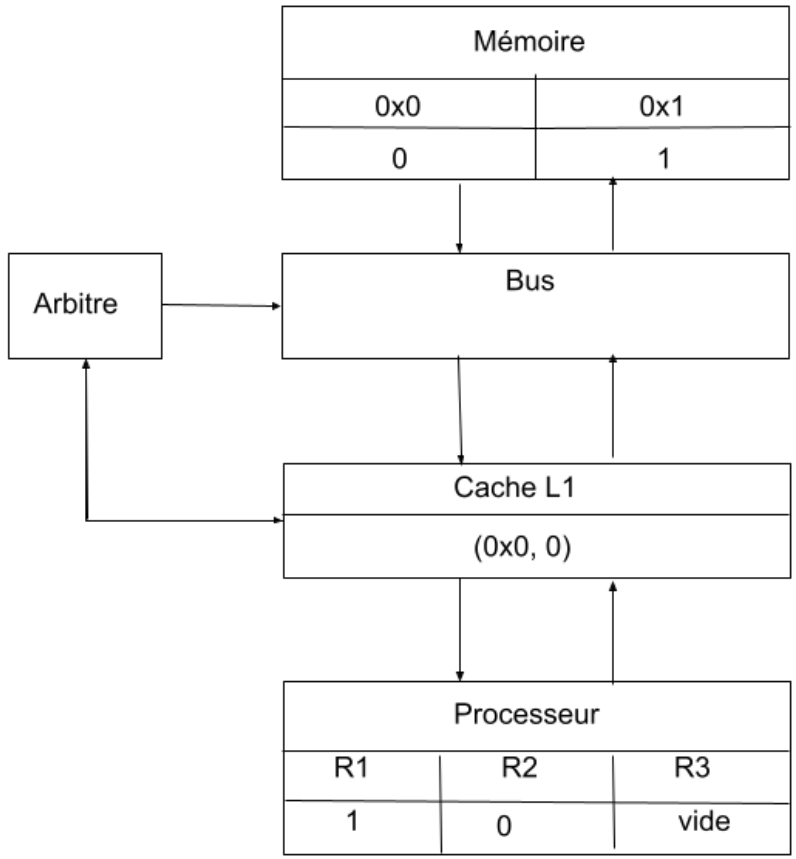


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

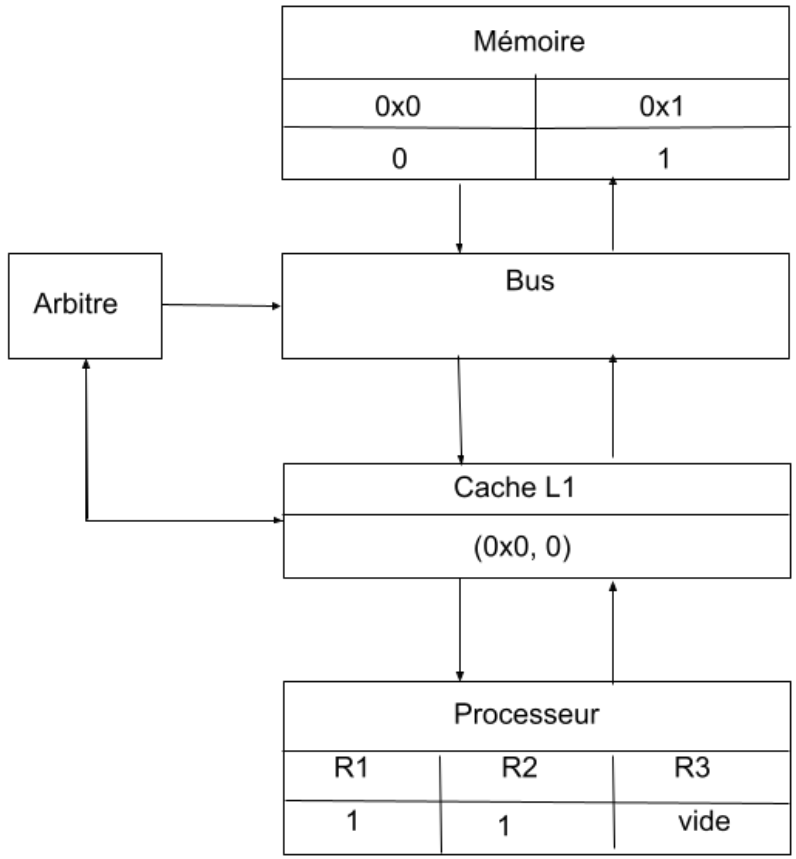
- ld R1, [0]
- add R1, R1, 1
- st R1, [1]
- ld R2, [0]



Cas monoprocesseur : effet cache

P1a :

- ld R1, [0]
- add R1, R1, 1
- st R1, [1]
- ld R2, [0]
- add R2, R2, 1

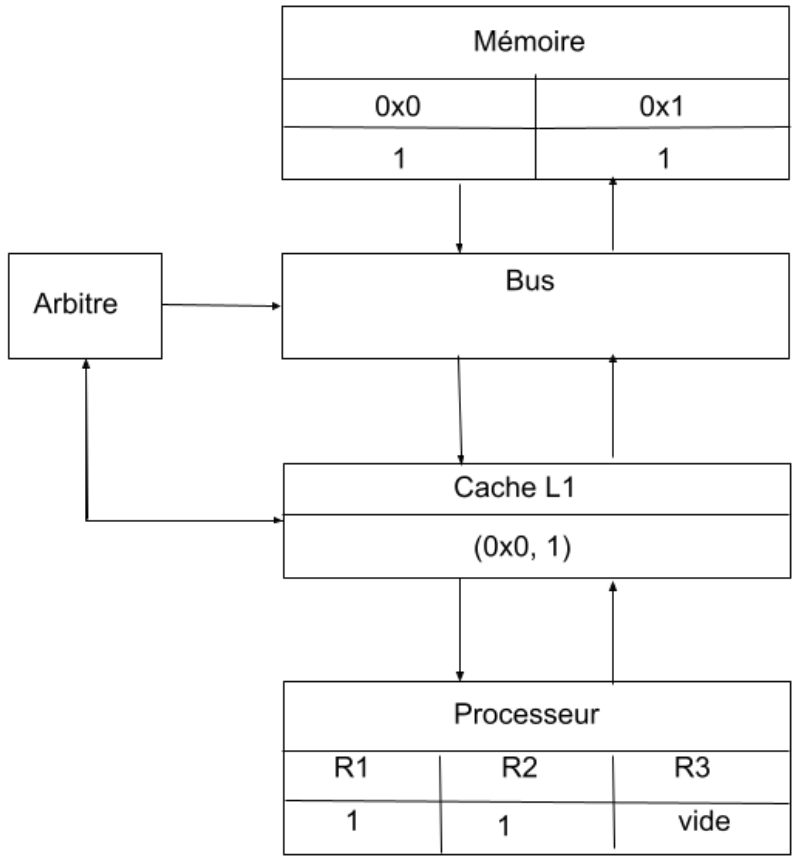


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : effet cache

P1a :

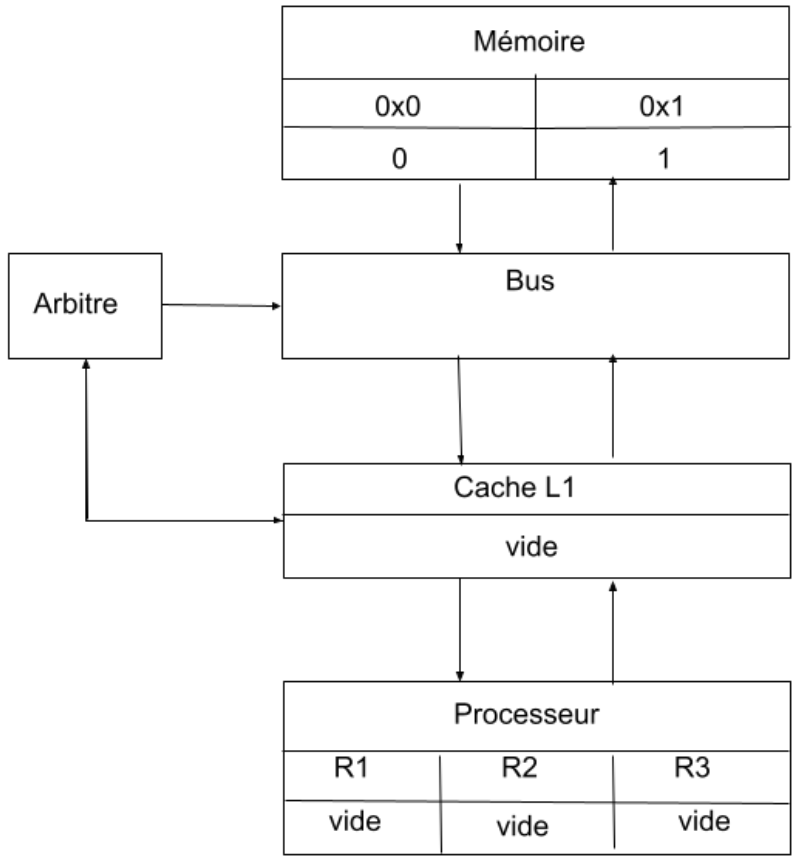
- ld R1, [0]
- add R1, R1, 1
- st R1, [1]
- ld R2, [0]
- add R2, R2, 1
- st R2, [0]



Etude du protocole et des accès aux données partagées

Cas monoprocesseur : éviction du cache / écriture write-trough

P1b :

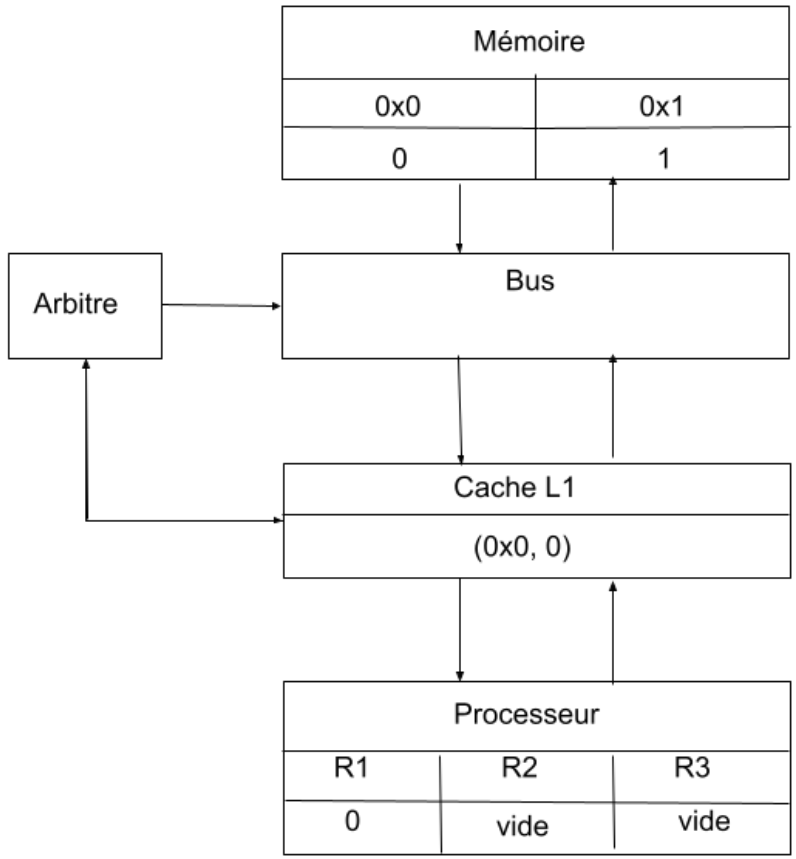


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : éviction du cache / écriture write-trough

P1b :

- ld R1, [0]

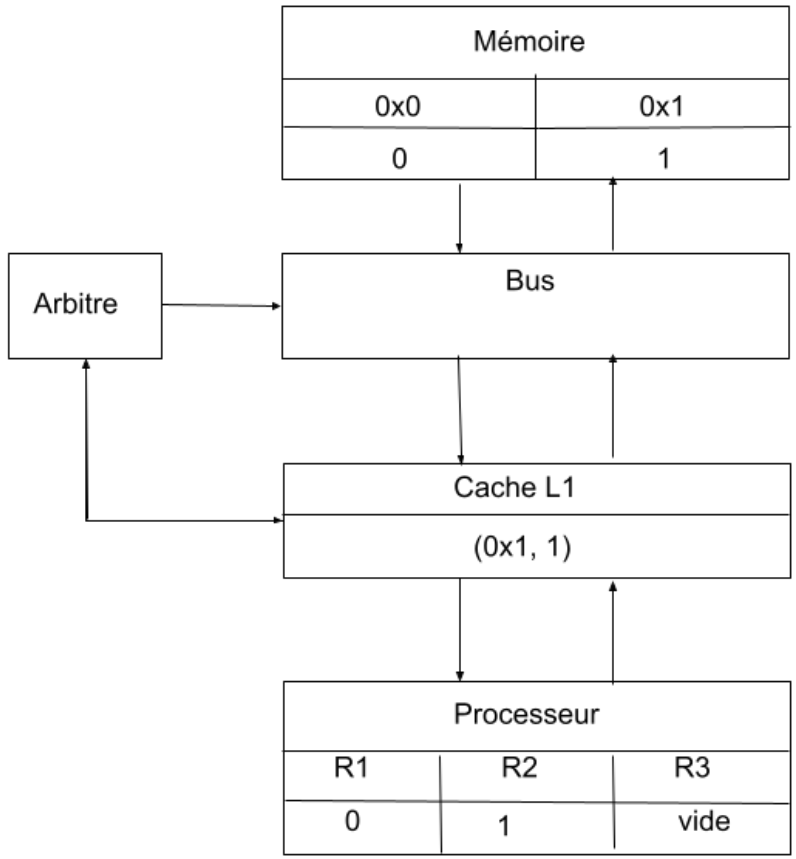


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : éviction du cache / écriture write-trough

P1b :

- ld R1, [0]
- ld R2, [1]

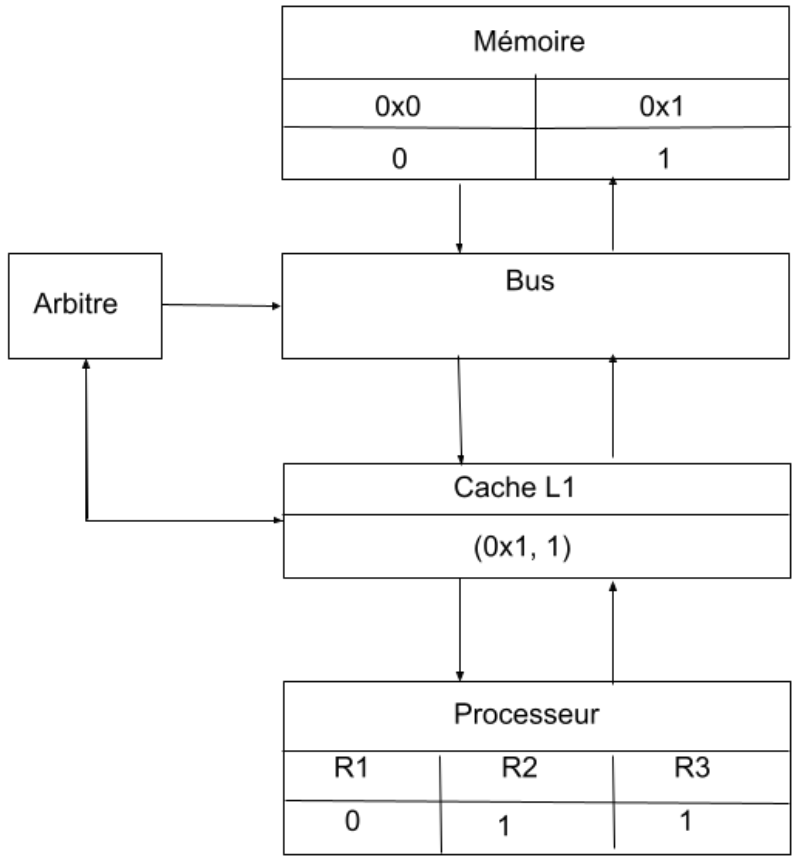


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : éviction du cache / écriture write-trough

P1b :

- ld R1, [0]
- ld R2, [1]
- add R3, R1, R2

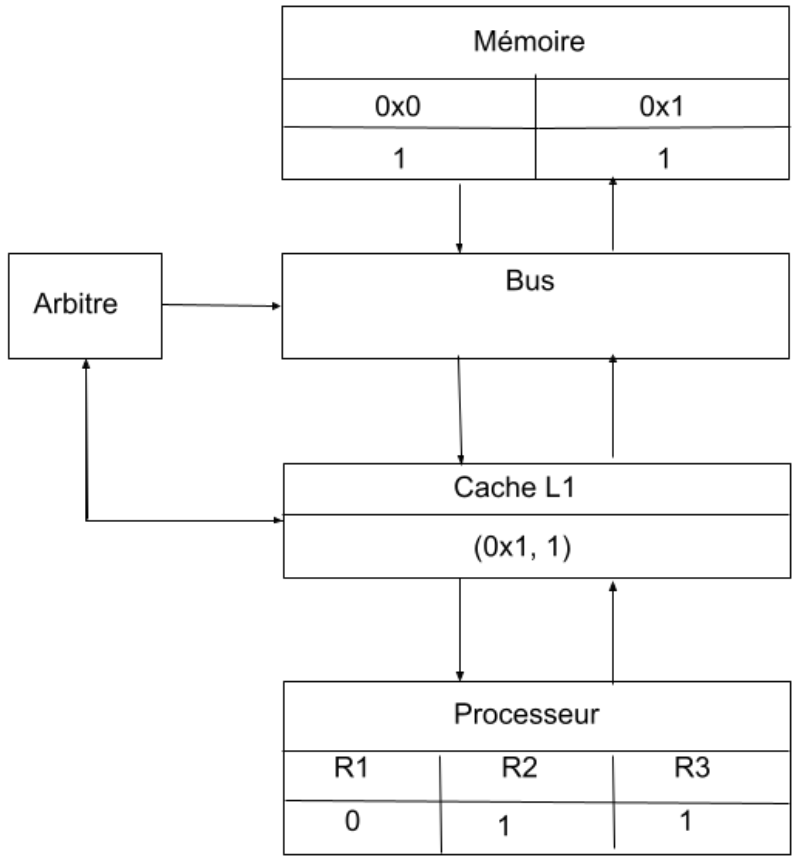


Etude du protocole et des accès aux données partagées

Cas monoprocesseur : éviction du cache / écriture write-trough

P1b :

- ld R1, [0]
- ld R2, [1]
- add R3, R1, R2
- st R3, [0]



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : partage en lecture / une écriture

P1a :

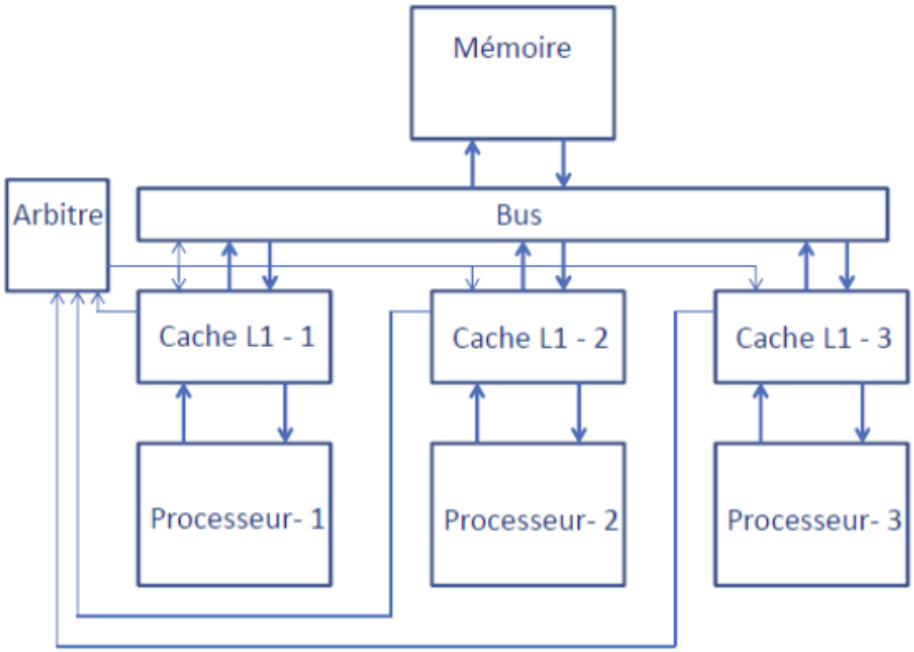
- ld R1, [0]

P2a :

- ld R1, [0]

P3a :

- inactif



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : partage en lecture / une écriture

P1a :

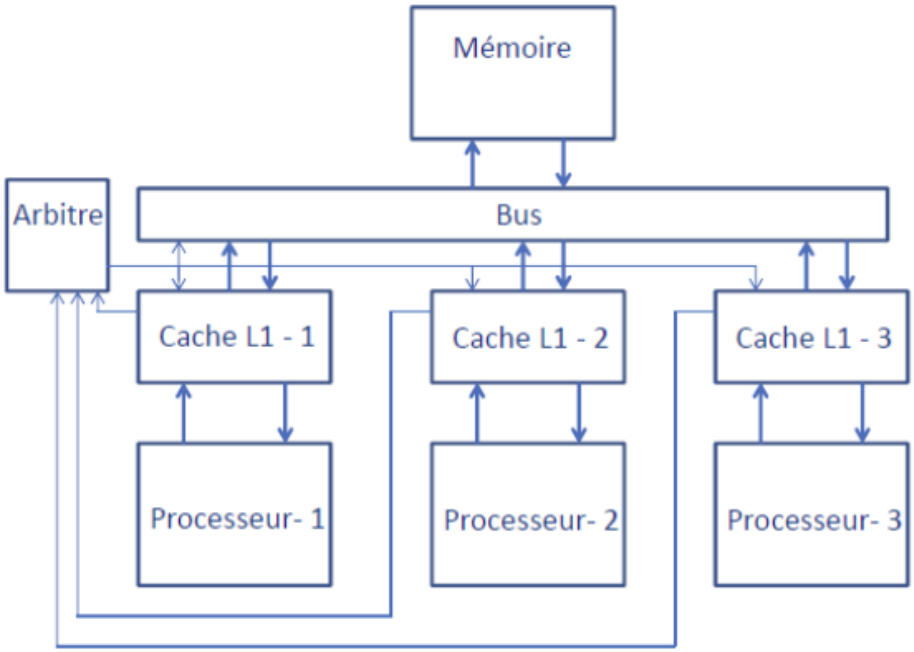
- add R1, R1, 1

P2a :

- ...

P3a :

- inactif



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : partage en lecture / une écriture

P1a :

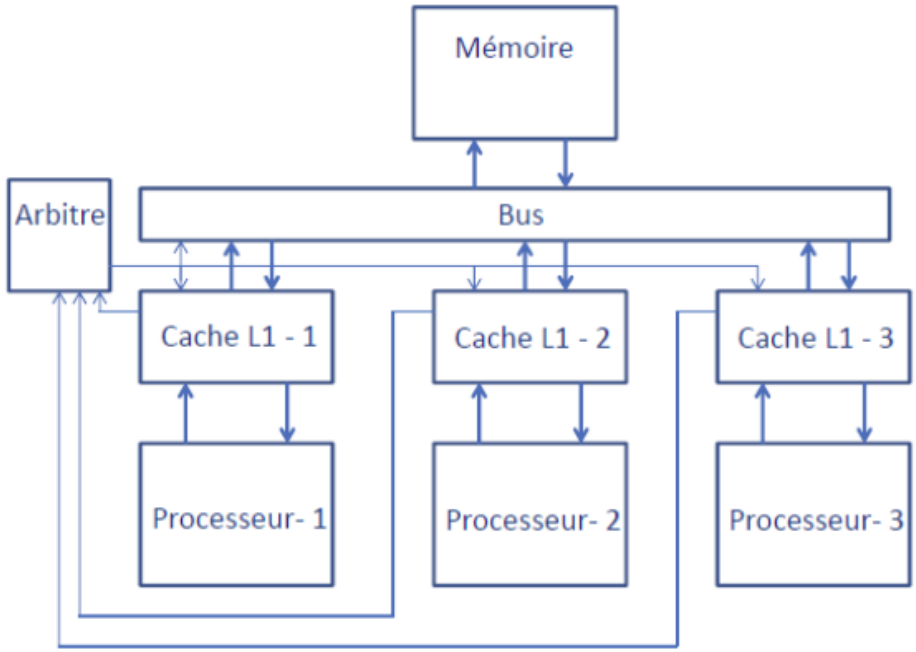
- st R1, [0]

P2a :

- ...

P3a :

- inactif



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : partage en lecture / une écriture

P1a :

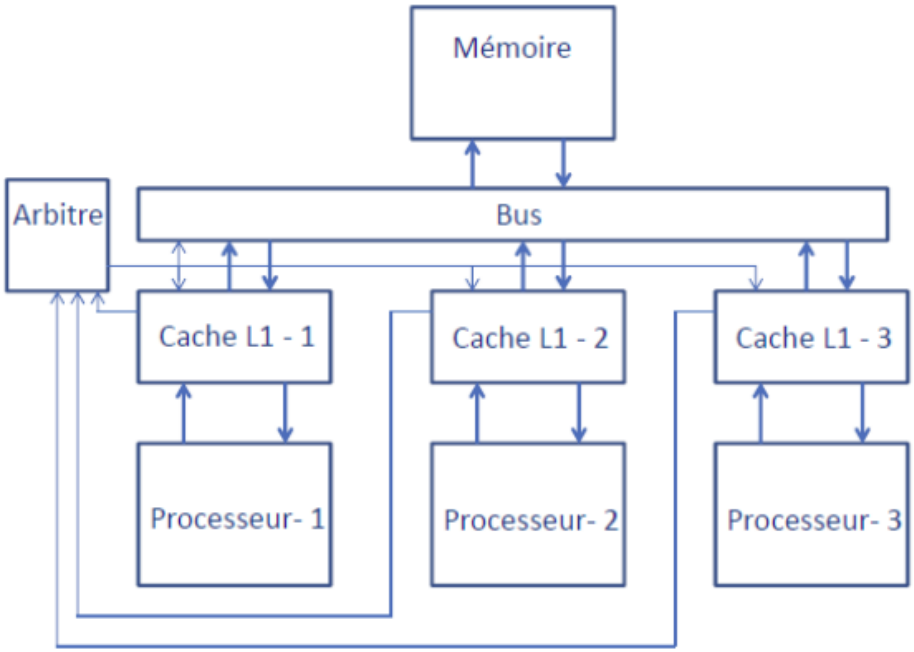
- ...

P2a :

- add R2, R1, 1

P3a :

- inactif



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : partage en lecture / une écriture

P1a :

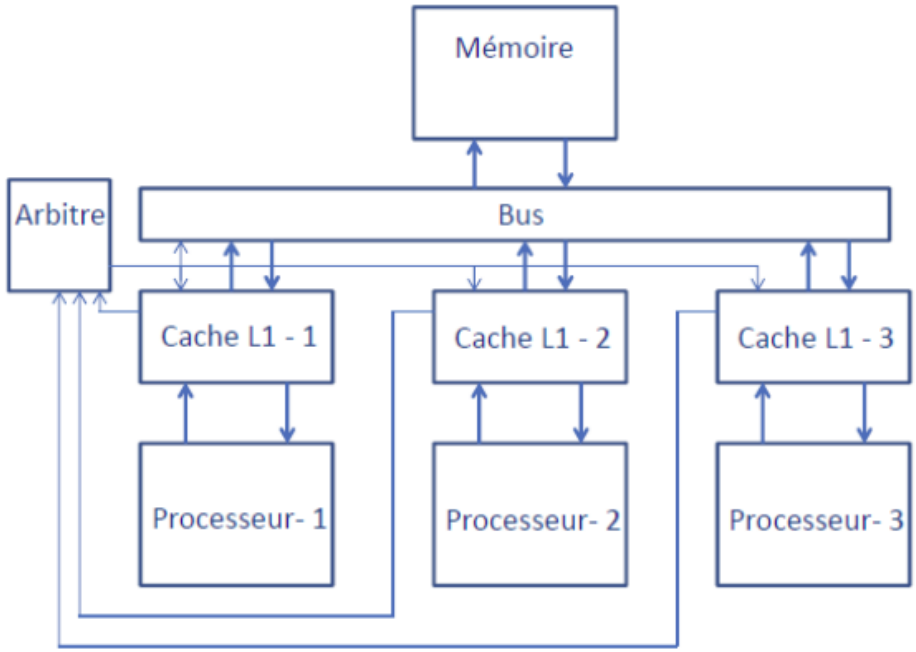
- ...

P2a :

- st R2, [1]

P3a :

- inactif



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

P1b :

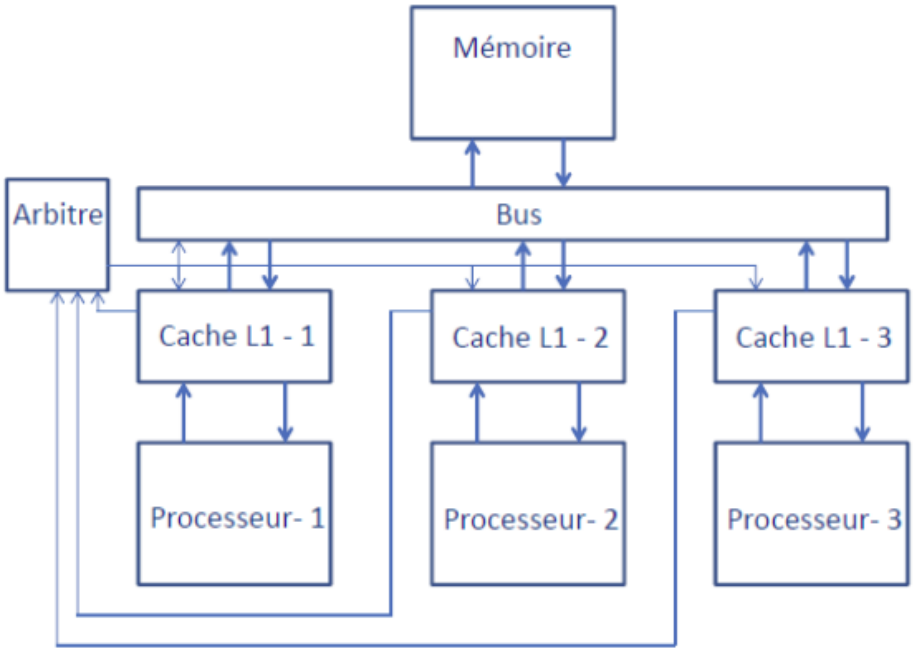
- dbt : ld R1, [0]

P2b :

- dbt : ld R1, [0]

P3b :

- ...



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

P1b :

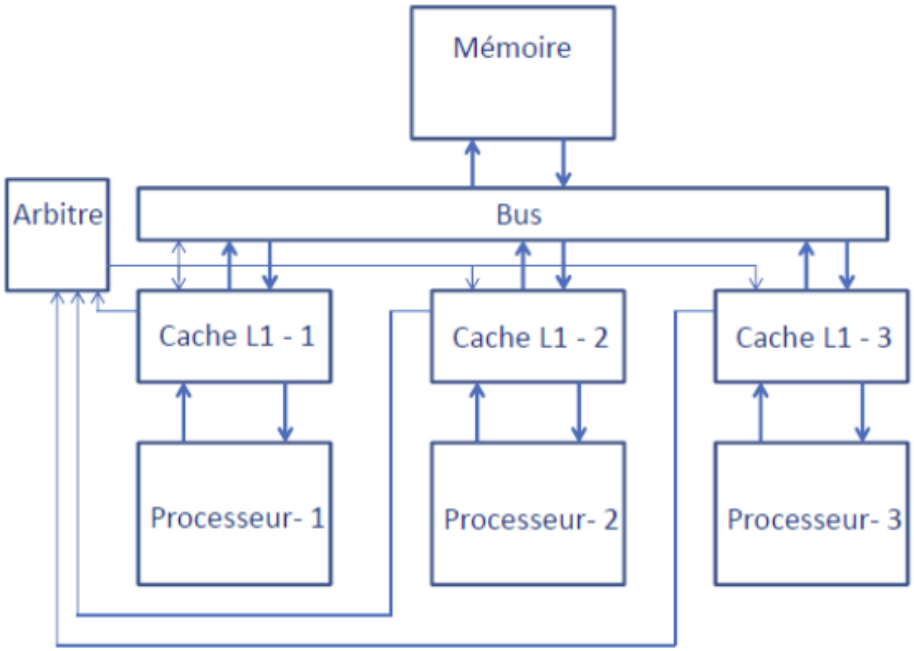
- cmp R1, 0

P2b :

- cmp R1, 0

P3b :

- ...



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

P1b :

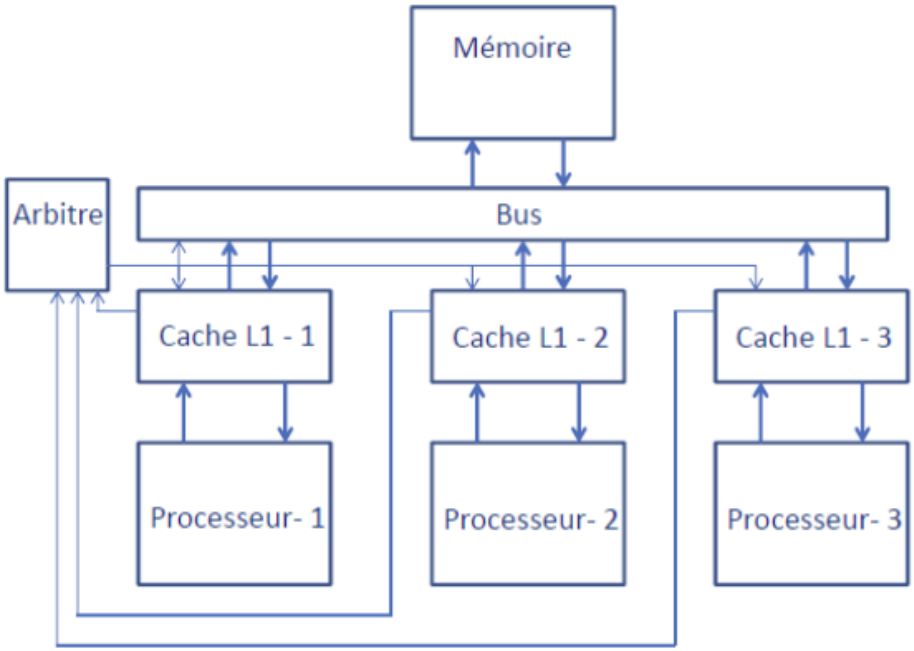
- beq dbt

P2b :

- beq dbt

P3b :

- ...



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

P1b :

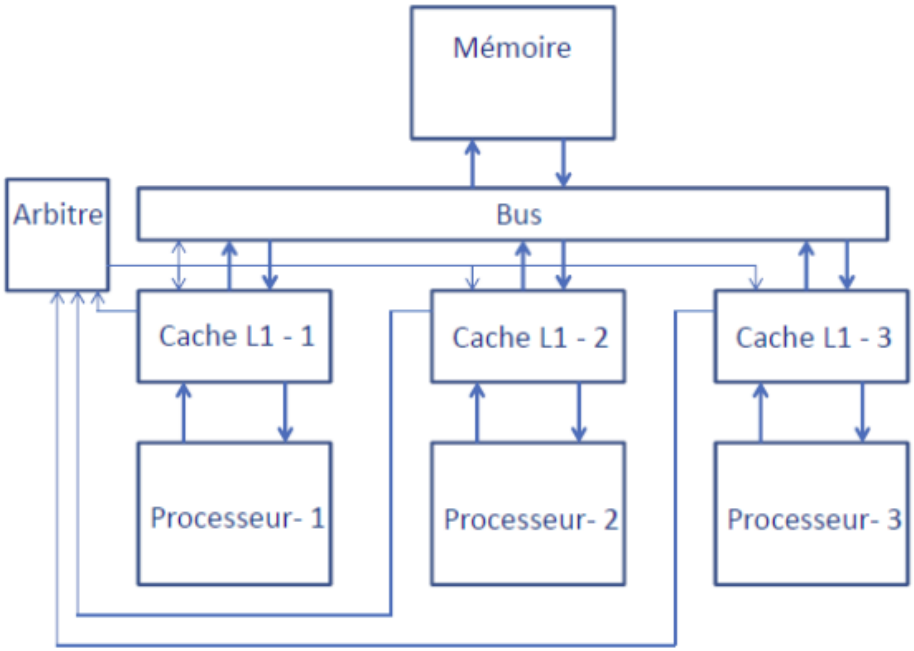
- dec R1

P2a :

- dec R1

P3a :

- ld R1, [1]



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

P1b :

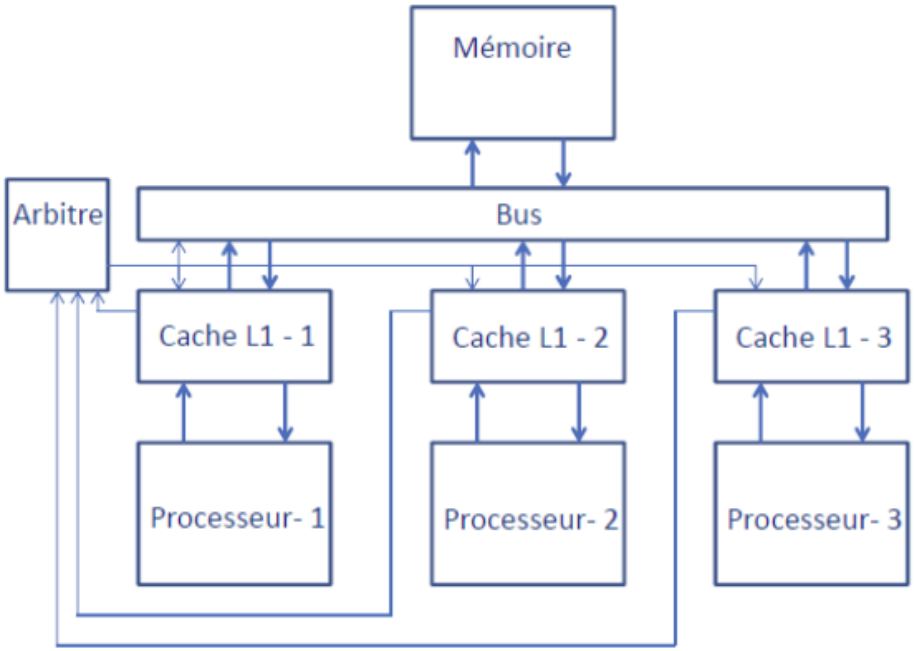
- st R1, [0]
- sc

P2b :

- st R1, [0]
- sc

P3b :

- ...



Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

Mécanisme de SNOOP

Lorsqu'une requête d'écriture circule sur le bus, chaque cache connecté doit déterminer s'il dispose du mot concerné. Dans l'affirmative, il met à jour sa copie locale avec la donnée à écrire, véhiculée sur le bus.

Etude du protocole et des accès aux données partagées

Cas multiprocesseur : boucle d'attente active / partage en lecture - écriture

Mécanisme de SNOOP

Lorsqu'une requête d'écriture circule sur le bus, chaque cache connecté doit déterminer s'il dispose du mot concerné. Dans l'affirmative, il met à jour sa copie locale avec la donnée à écrire, véhiculée sur le bus.

Garantir un accès exclusif aux données partagées

Un vérrou pour l'accès à une donnée.

Modélisation des composants

Architecture focalisée sur les interfaces de communication :
(schéma)

Modélisation des composants

Memory

```
MODULE Memory(bus_cmd, bus_data, bus_addr)
VAR data : array 0..1 of boolean;
ASSIGN
    init(data[0]) := FALSE;
    init(data[1]) := FALSE;
    next(data[0]) :=
        case
            (bus_addr = FALSE) & (bus_cmd = Wr) : bus_data;
            TRUE :          data[0];
        esac;
    next(data[1]) :=
        case
            (bus_addr = TRUE) & (bus_cmd = Wr) : bus_data;
            TRUE :          data[1];
        esac;
```

Modélisation des composants

Arbiter

```
MODULE Arbiter(L1_1_req)
VAR
    arb_gnt : boolean;
ASSIGN
    init(arb_gnt) := FALSE;
    next(arb_gnt) :=
        case
            L1_1_req & !arb_gnt := TRUE;
            TRUE      := TRUE;

--END Arbiter
```

Modélisation des composants

Processor

```
MODULE Processor(cache_busy, cache_data)
VAR
    mem_req :        {idle, Ld, St};
    eff_addr :     boolean;
    register :     boolean;

ASSIGN
init(mem_req) :=      idle;
next(mem_req) :=
    case
        !cache_busy : {idle, Ld, St};
        TRUE    : mem_req;
    esac;
```

Modélisation des composants

Processor (suite)

```
init(eff_addr):= FALSE;
next(eff_addr):=
    case
        !cache_busy : {FALSE,TRUE};
        TRUE    : eff_addr;
    esac;

init(register):= FALSE;
next(register):=
    case
        !cache_busy & (mem_req = St) : register;
        !cache_busy & (mem_req = Ld) : cache_data;
        TRUE         : {FALSE,TRUE};
    esac;
```

Merci de votre attention !