

M1 SESI 2017-2018  
Architecture Multi-Processeurs  
Notes de Cours

Alain Greiner  
*rédigé par Kevin Mambu*

April 11, 2018

# Contents

<b>Introduction</b>	<b>3</b>
<b>Automates d'Etats Finis Communiquants Synchrones</b>	<b>3</b>
Automates de Mealy vs Automates de Moore . . . . .	4
<b>Méthode Générale de Synthèse des FSM</b>	<b>5</b>
Exemple : capteur de trois '1' consécutifs, MOORE . . . . .	6
<b>Bus Système : Le PIBUS</b>	<b>8</b>
<b>Fonctions d'un bus</b>	<b>8</b>
<b>Caches</b>	<b>11</b>
Caches de premier niveau L1 . . . . .	11
Principes des mémoire caches . . . . .	11
Vocabulaire . . . . .	11
Spécifications du cache L1 vu en cours . . . . .	12
Spécifications des Interfaces 1 . . . . .	13

# Introduction

Responsable : Alain Grener, alain.grener@lip6.fr

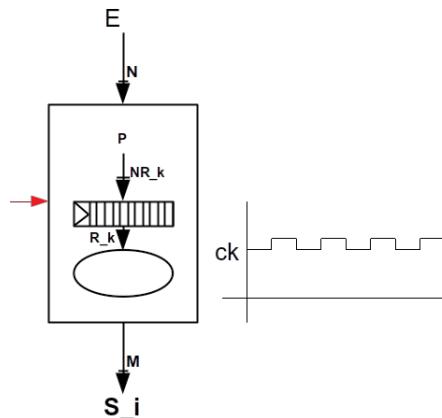
Problématique de l'UE : synchronisation de plusieurs composants dans une architecture multi-processeur. Exemple : accès du même périphérique par plusieurs processeurs. Observer le moyen de communication entre processeurs et l'environnement des processeurs. Un processeur pipeline reste synchronisé : plusieurs instructions sur le même flux de transition. Dans le cas d'une architecture multi-processeurs  $\Rightarrow$  les unités sont nativement asynchrones, mais doivent se synchroniser.

## Automates d'Etats Finis Communiquants Synchrones

Il s'agit d'un modèle mathématique, qui permet de décrire tous les systèmes matériels numériques. "Les états" ici est l'ensemble des valeurs intégrées à l'automate des ensembles mémorisants. Un état est une valeur stockée dans les registres.

Exemple avec le processeur MIPS32 : comptons le banc de registres, PC, HI, LO, etc. On peut comptabiliser environ 50 registres, dans ce cas le processeur MIPS32 possède  $2^{50}$  états.

À chaque front montant, on met à jour l'ensemble des registres internes. Les sorties d'un automate sont les entrées d'un autre. Un automate produit est l'ensemble des sous-systèmes (automates).



$$S_i = g_i(E_0, E_1, \dots, E_{(N-1)}, R_0, R_1, \dots, R_{(P-1)}, M)$$

M fonctions booléennes dépendant de  $N + P$  bits.

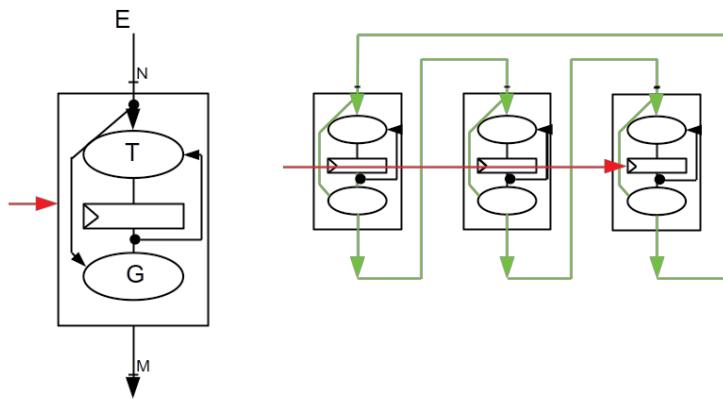
$$NR_k = T_k(E_0, E_1, \dots, E_{(N-1)}, R_0, R_1, \dots, R_{(P-1)}, P)$$

P fonctions booléennes dépendant de  $N + P$  bits.

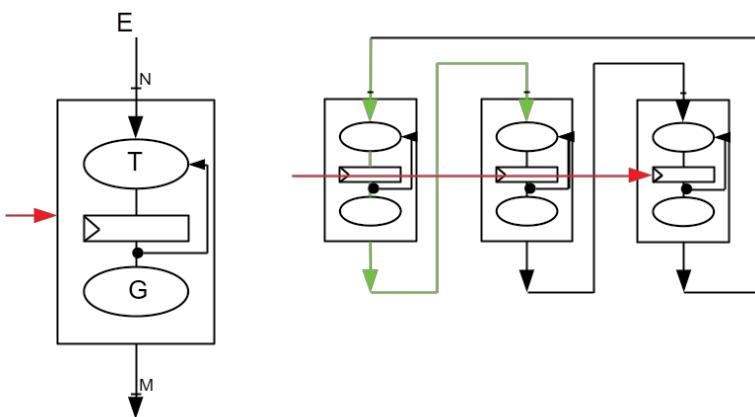
Pour décrire complètement le comportement d'un automate particulier, il faut déterminer:

1. quelles valeurs vont prendre les sorties en fonction des entrées  $E_i$  et de l'état.  $\Rightarrow$  décrire un système séquentiel (à mémoire).
2. les valeurs de l'état suivant  $NR_k$  en fonction des entrées  $E_i$  et de l'état courant  $R_k$ .
3. un mécanisme permettant d'initialiser l'état interne  $\Rightarrow$  état initial  $R_k^0$ .

## Automates de Mealy vs Automates de Moore



La génération de la sortie dépend de l'état de transition et des entrées



La génération de la sortie dépend uniquement de l'état de transition

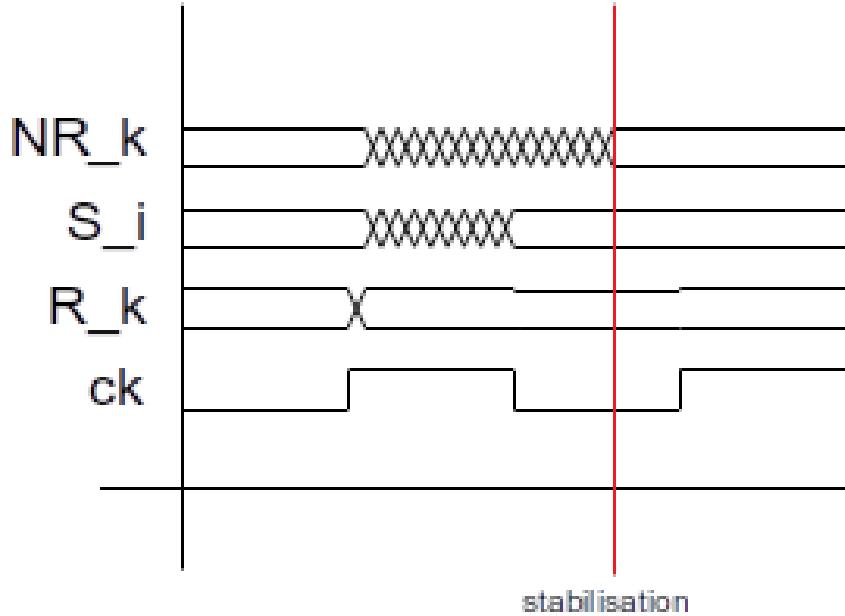
Avantages de l'Automate de Moore :

- Facilement prédictible (synchronisation assurée)
- Beaucoup plus stable sur la propagation de l'information

Inconvénients :

- Moins réactif

Le plus long chemin possible est le temps de cycle minimal d'un registre à un autre.



La génération de la sortie dépend uniquement de l'état de transition

## Méthode Générale de Synthèse des FSM

Toute l'industrie de la CAO repose sur la technologie des FSM.

⇒ Par extension, toute l'industrie du High-Tech.

### ETAPE A:

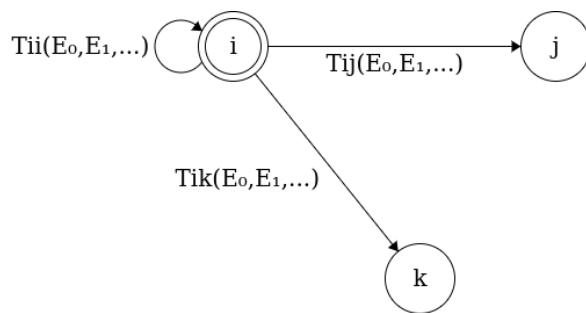
Définition des "états abstraits". Il s'agit d'identifier les états avec des symboles et faire abstraction de leur description pour leur nomination.

### ETAPE B:

Représentation abstraite du graphe de transition.

Il va s'agir d'un graphe orienté dont on doit précisément spécifier :

- Les états (un noeud par état accessible)
- Les transitions (les arcs orientés) :



On doit s'assurer des conditions d'orthogonalité et de complétude :

- orthogonalité ⇒ 2 transitions de sortie ne peuvent pas être simultanément accessibles :

$$\forall E_i \forall i \forall j \forall k (T_{ij} \bullet T_{ik}) \equiv 0 \text{ si } i \neq j$$

- complétude ⇒ Il y a toujours une transition de sortie qui est vraie :

$$\forall E_i \forall i \sum j = 0(T_{ij}) = 1$$

Nb : Les états sont étiquetés en fonction de fonction booléenne de l'expression de la valeur de sortie

**ETAPE C:**

Choix d'un codage des états.

**ETAPE D:**

Traduction du graphe en table de vérité.

**ETAPE E:**

Déterminer les équations booléennes simplifiées.

**ETAPE F:**

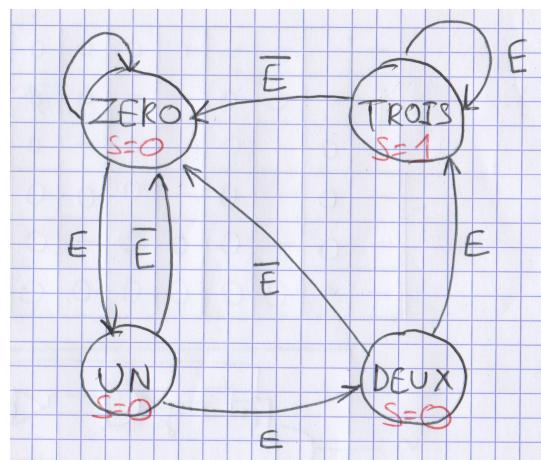
Traduction en portes logiques

**Exemple : capteur de trois '1' consécutifs, MOORE**

Etape A :

- "ZERO", aucun chiffre mémorisé
- "UN", le dernier caractère détecté est 1
- "DEUX", les deux derniers caractères détectés sont '1' et '1'
- "TROIS", les trois derniers caractères détectés sont '1', '1' et '1'

Etape B :



Etape C :

	$R_1$	$R_0$	$R_3$	$R_2$	$R_1$	$R_0$
ZERO	0	0	0	0	0	1
UN	0	1	0	0	1	0
DEUX	1	0	0	1	0	0
TROIS	1	1	1	0	0	0

NBC                    one-hot

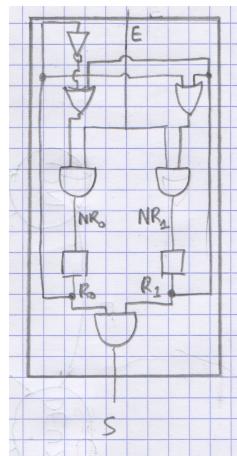
Etape D

$R_1$	$R_0$	$E$	$NR_1$	$NR_0$	$S$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Etape E

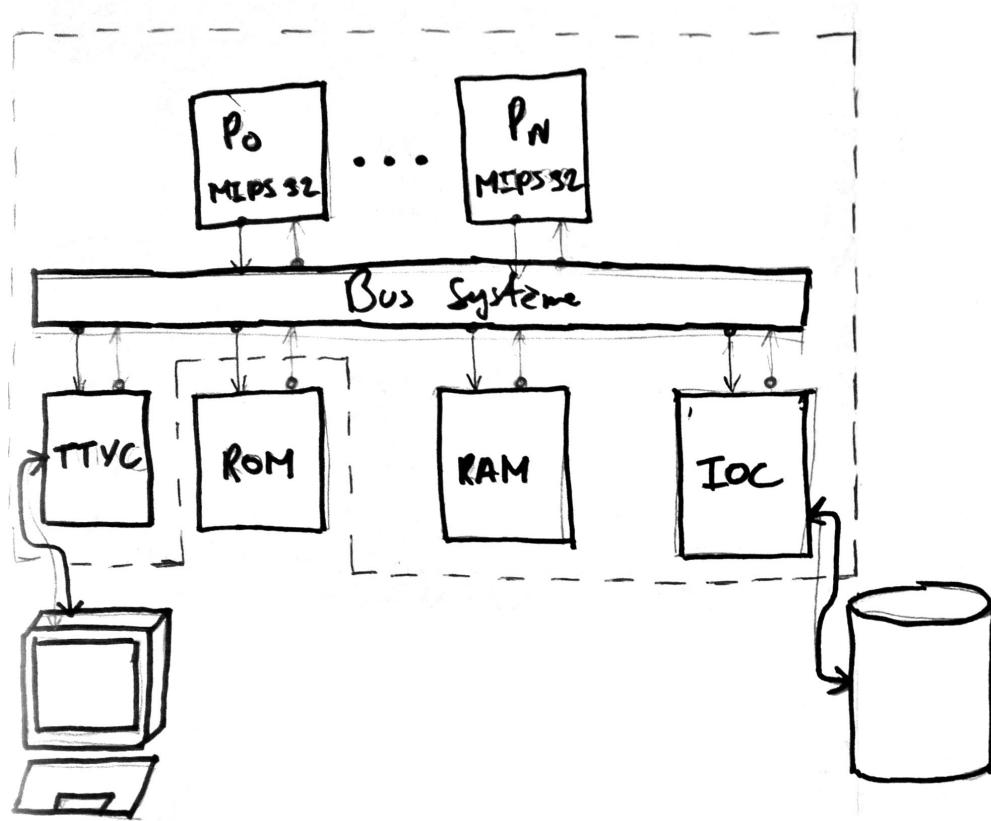
- $S = R_1 \bullet R_0$
- $NR_1 = E \bullet (R_0 + R_1)$
- $NR_1 = E \bullet (\overline{R_0} + R_1)$

Etape F

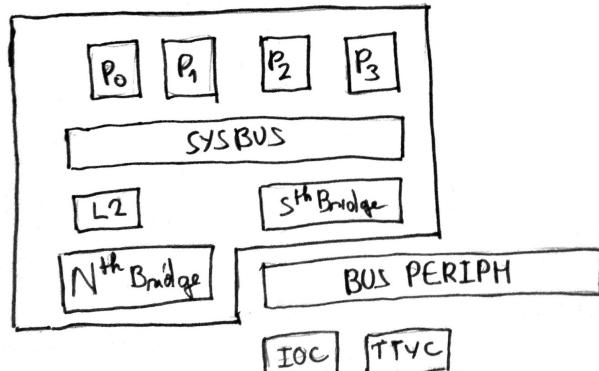


## Bus Système : Le PIBUS

Le bus permettant d'interconnecter les processeurs avec les autres composants.



De nos jours, le bus ne se contente plus d'être une simple nappe de fils, il s'agit d'un réseau d'interconnection sur puce. Le bus est devenu un composant crucial des machines.



Le North Bridge fait transiter les données vers la RAM, le South Bridge vers le bus périphérique. Tous deux sont des contrôleurs sur puce.

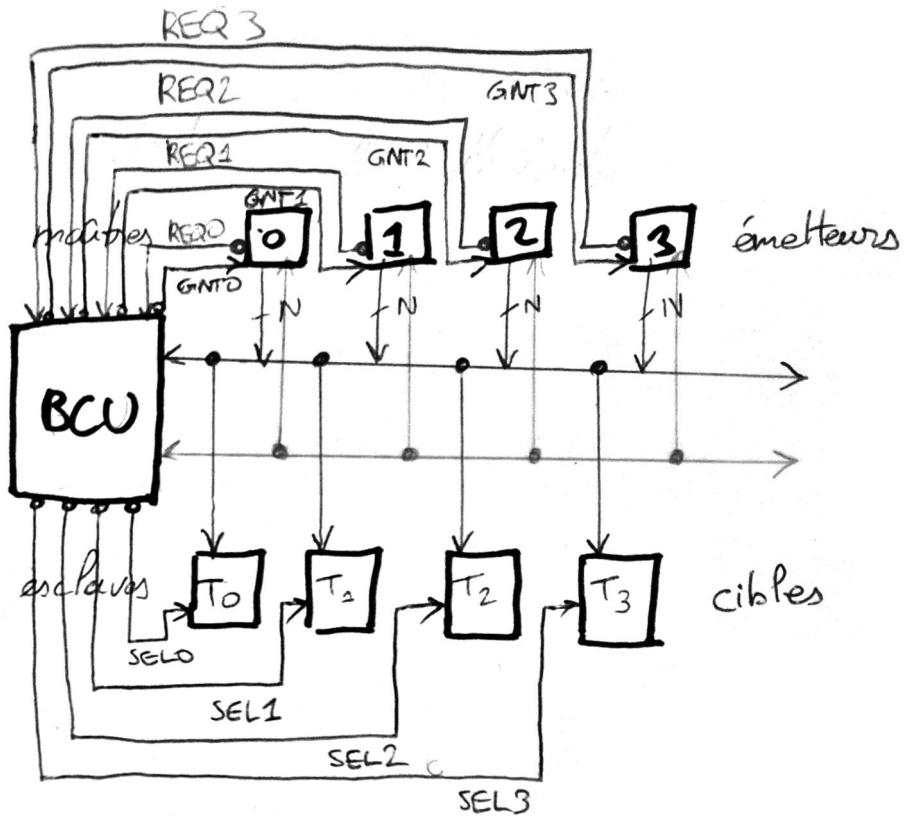
## Fonctions d'un bus

1. Acheminement des commandes des maîtres vers les bonnes cibles :

- routage vers la bonne cible
- routage vers les différents maîtres

2. Acheminement des réponses des cibles vers le bon maître

- arbitrage entre les cibles



**transaction** : paire d'une commande d'un maître et d'une réponse de la cible.  
Cela est vrai pour les lectures **et** les écritures, et ce afin de :

- notifier d'éventuelles erreurs, et permettre un debogage plus exhaustif.
- **prévenir de problèmes de synchronisation**

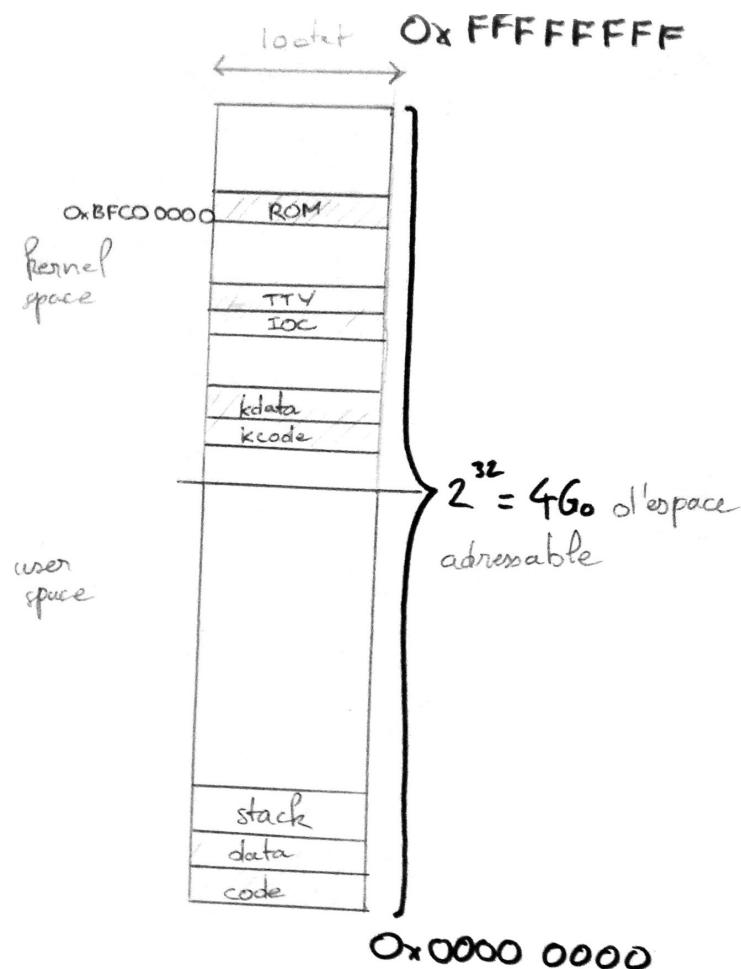
Exemple :

Deux applications multithreadées communicantes par une case mémoire : système producteur/consommateur. Le producteur doit notifier le consommateur à la fin de sa routine → utilisation d'une deuxième case mémoire de synchronisation.

D'abord produire dans la première case, puis notifier dans la deuxième.

Le bus a pour particularité de ne pouvoir faire qu'une transaction à la fois, cela permet au bus d'uniquement enregistrer l'émetteur pour le routage de la réponse cible.

Pour une définition plus basse, un bus est un signal connecté à plusieurs émetteurs où un et un seul de ces émetteurs est non-bloqué à la fois. On parle d'un **signal bus'é**.



# Caches

## Caches de premier niveau L1

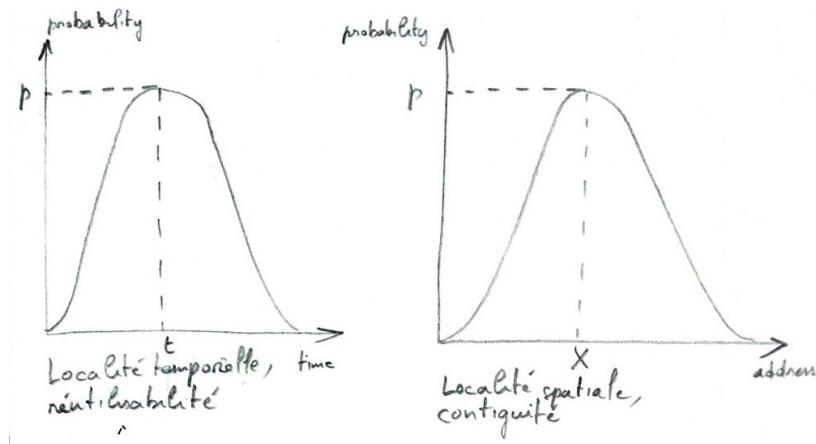
Au cours précédent, nous avons étudié le comportement d'un automate pré-programmé qui effectuait machinalement des tâches d'opérations et de requêtes à un ou plusieurs composants cibles.

Néanmoins, dans le cas d'un automate reprogrammable comme le MIPS32, utilise des intermédiaires pour ses accès mémoires en l'objets de caches. Plus spécifiquement de contrôleurs de caches, afin d'éviter une temporisation du processeur.

## Principes des mémoire caches

La performance des caches est basée sur deux principes intrinsèques à la nature des données en mémoire :

- La localité spatiale, soit deux adresses X et X', après lecture de X, plus X' est proche de X, plus sa probabilité d'être la prochaine adresse requise est élevée. La localité spatiale a pour critère principale la contiguïté des données.  
Ex: lecture d'un tableau, lecture (souvent [toujours]) séquentielle des instructions.
- La localité temporelle, soit une adresse X, après lecture de X, plus de temps s'écoule entre cet accès et le prochain, et plus sa probabilité d'être encore la prochaine adresse requise est basse. La localité temporelle a pour critère principale la réutilisabilité des données.  
Ex: exécution d'une boucle d'instruction



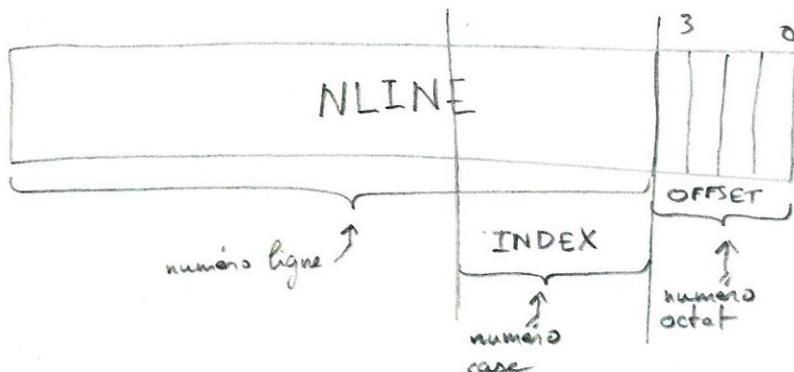
## Vocabulaire

- **Alignement** : une portion de la mémoire est dite alignée si et seulement si son adresse de base est multiple de la longueur de la portion.

- **Ligne de cache** :

nombre fixe d'octets à des adresses consécutives et alignées (multiples de 4). Elle représente une portion de l'espace adressable.

Sur une architecture MIPS32 l'espace adressable est partitionné en  $\frac{2^{32}}{2^4} = 2^{28} = 2^{10} \times 2^{10} \times 2^8 = 256$  million de lignes.  
Nous considérerons des lignes de caches de 16 octets :

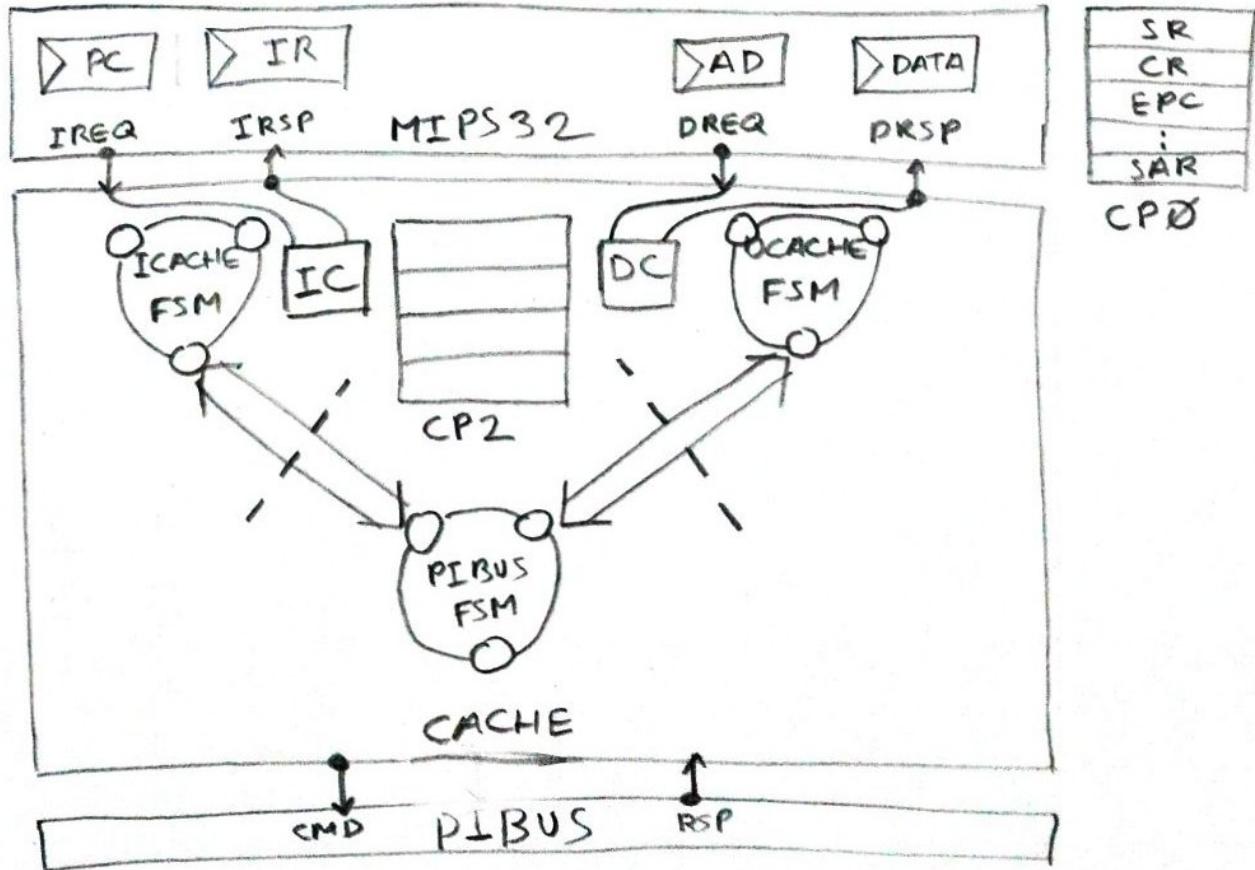


- **Cache à Correspondance Directe :**  
Une ligne de cache ne peut avoir qu'une case prédéfinie dans la mémoire cache.
- **Cache Associatif :**
  - *Associatif Complet* :  
Toute ligne du cache peut être stockée dans n'importe quelle case dans la mémoire cache.
  - *Associatif par Ensemble* :  
Une ligne de cache peut avoir un certain nombre de cases prédéfinies dans la mémoire cache, selon le degré d'association.
- **Politique Write-Through :**  
Toute requête d'écriture du processeur est envoyée à la mémoire, sans attente, sur le bus. Le Tampon d'Écritures Postées permet au processeur de ne pas être bloqué sous réserve de tampon non-plein.
- **Politique Write-Back :**  
Toute écriture d'une adresse X de la part du processeur est effectuée dans le cache L1. S'il y a MISS, alors le processeur requiert que la ligne concernant X soit ramenée dans le cache.
- **Tampon d'Écriture Postée :**  
Le TEP est un composant permettant de mettre en attente plusieurs requêtes vers le PIBUS sans pour autant geler le processeur (sous réserve du tampon non-plein).  
Un mécanisme d'implémentation parmi d'autres du TEP est le protocole FIFO.

## Spécifications du cache L1 vu en cours

- 2 caches séparés pour instructions et données :  
Cela permet un comportement pipeline, avec les émissions de requêtes de lectures d'instructions et d'accès mémoire possible
- Politique Write-Through :  
simplifie les problèmes de cohérence
- Associatif par ensemble paramétrable :
  - NWORDS : nombre de mots par lignes
  - NSETS : nombre d'ensembles
  - NWAYS : degré d'associativité (NWAYS = 1  $\Rightarrow$  Correspondance Directe).
  - $NWAYS \times NSETS \times NWORDS = \#Cache$
- **Bloquant :**  
Tant que le cache n'a pas acquitté de la requête du processeur, ce dernier est bloqué (parce que le MIPS32 est pipeline scalaire, ce n'est pas le cas pour les processeurs superscalaires par exemple).

# Spécifications des Interfaces 1



- **IREQ :**

- addr(30) : 30 bits de poids forts car alignée (multiple de 4)
- valid(1) : envoi ou non de la requête
- mode(1) : utilisateur ou système

- **IRSP :**

- instr(32)
- valid(1) : bloque le processeur si égal à 0
- error(1) : prévient d'une erreur de bus si égal à 1

- **DREQ :**

- addr(30) : 30 bits de poids forts car alignée (multiple de 4)
- valid(1)
- mode(1)
- type(3) : voir spécification du signal DTYP
- wdata(32)
- byte\_enable : spécifie les octets à écrire (codage one-hot)

- **DRSP :**

- rdata(32)

- valid(1)
  - error(1)
- **DYTPE :**
    - READ
    - Write
    - Linked\_Load
    - Store\_Cond
      - Opérations pour implémenter l'accès atomique à une ressource
    - XTN\_READ
    - XTN\_WRITE
      - Accès en dehors de l'espace adressable courant (ex : co-processeurs)