Mémoire cache

ARO

9 - Mémoire cache

Résumé du document

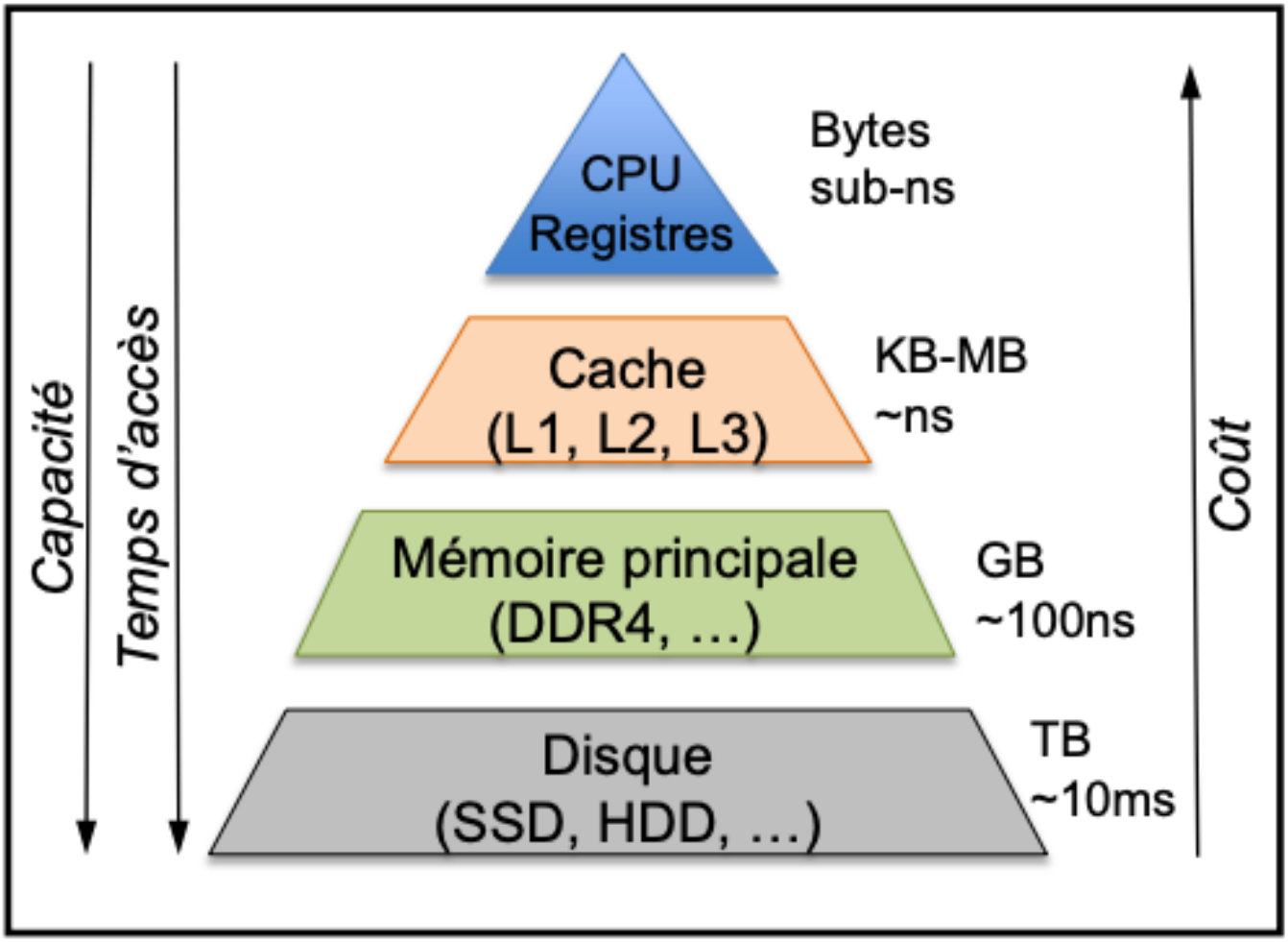
Dans ce document, nous allons aborder en détail la mémoire cache, un élément essentiel pour améliorer la performance des processeurs en stockant temporairement des données fréquemment utilisées. Nous expliquerons sa définition, son fonctionnement, sa structure interne, et ses différentes stratégies de placement. Nous discuterons également des concepts de localité temporelle et spatiale, et des niveaux de cache, tout en soulignant l'importance des succès (hits) et des échecs (misses) dans l'accès aux données.

Table des matières

1. Définition

La mémoire cache est une petite mémoire très rapide qui se trouve proche du processeur et permet de stocker des données fréquemment utilisées par le processeur. Elle permet d'accélérer l'accès aux données en les stockant temporairement à proximité du processeur.

C'est une mémoire qui semble à la fois grande et rapide et est transparante pour l'utilisateur.



1.1. Généralités

- Aussi appelée **antémémoire**
- Utilité : zone de mémoire plus rapide que la RAM où on stocke une copie d'informations qui se trouvent en mémoire RAM
- La mémoire cache est composée de transistors, et non de condensateurs
  - Même fréquence que le CPU
  - Latence de quelques cycles CPU
  - Aucun contrôle sur le contenu du cache : se remplit au fur et à mesure de l'exécution des programmes

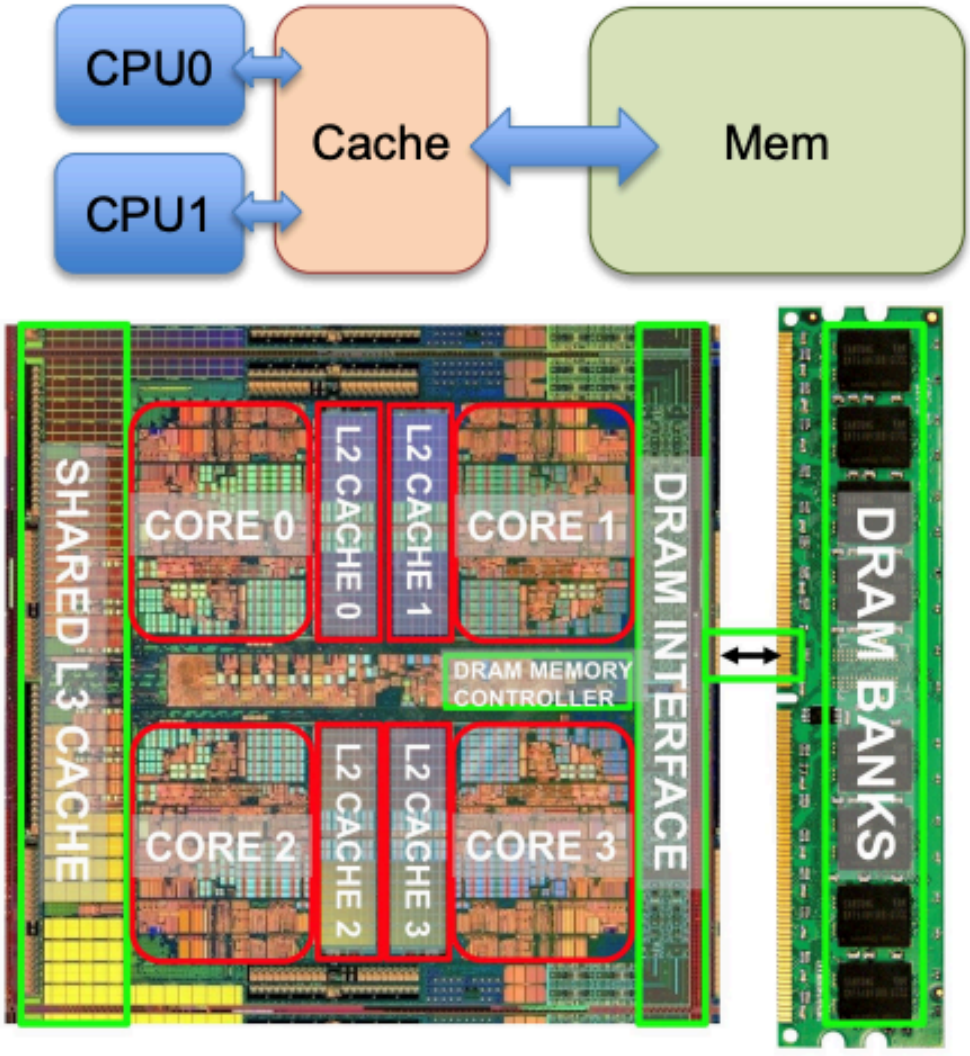
2. Fonctionnement

2.1. Niveau de stockage

La mémoire cache peut-être divisée en plusieurs niveaux comme L1, L2, L3, etc. Chaque niveau de cache est plus grand et plus lent que le précédent. Le cache L1 est le plus rapide et le plus petit, il est situé le plus près du processeur. Le cache L2 est plus grand et plus lent que le cache L1, et ainsi de suite.

Ensuite il y a la mémoire vive (mémoire principale) qui se trouve hors du processeur et qui est donc plus lente que la mémoire cache.

L'articulation des différentes mémoires peut se voir dans le schéma suivant :



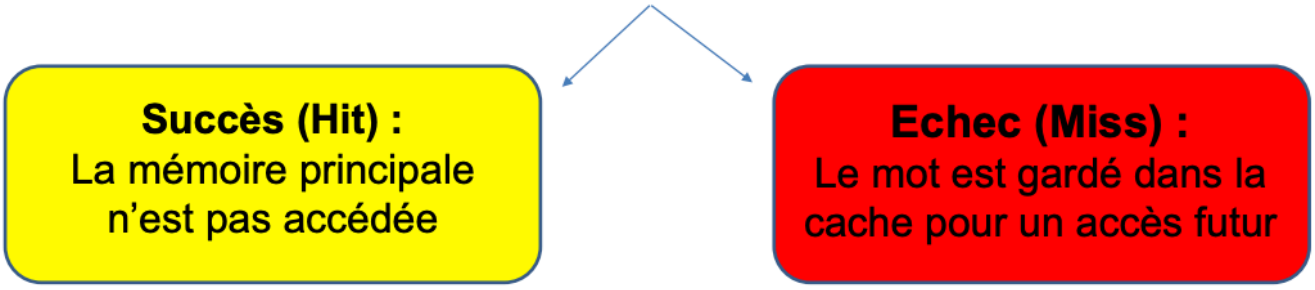
2.2. Objectif

- Utilité : zone de mémoire plus rapide que la RAM où on stocke une copie d'informations qui se trouvent en mémoire RAM
- La mémoire cache est composée de transistors, et non de condensateurs

3. Principe général

Le processeur essaie d'accéder un mot (octet) d'abord dans le cache. Si le mot est trouvé dans le cache, on parle de **hit** et le mot est renvoyé au processeur. Si le mot n'est pas trouvé dans le cache, on parle de **miss** et le mot est cherché dans la mémoire principale. Le mot est alors copié dans le cache et renvoyé au processeur.

Pour réduire le temps de mise à jour lors d'un échec, le **bus** entre mémoire et cache est plus large que celui entre cache et processeur.



3.1. Localité temporal

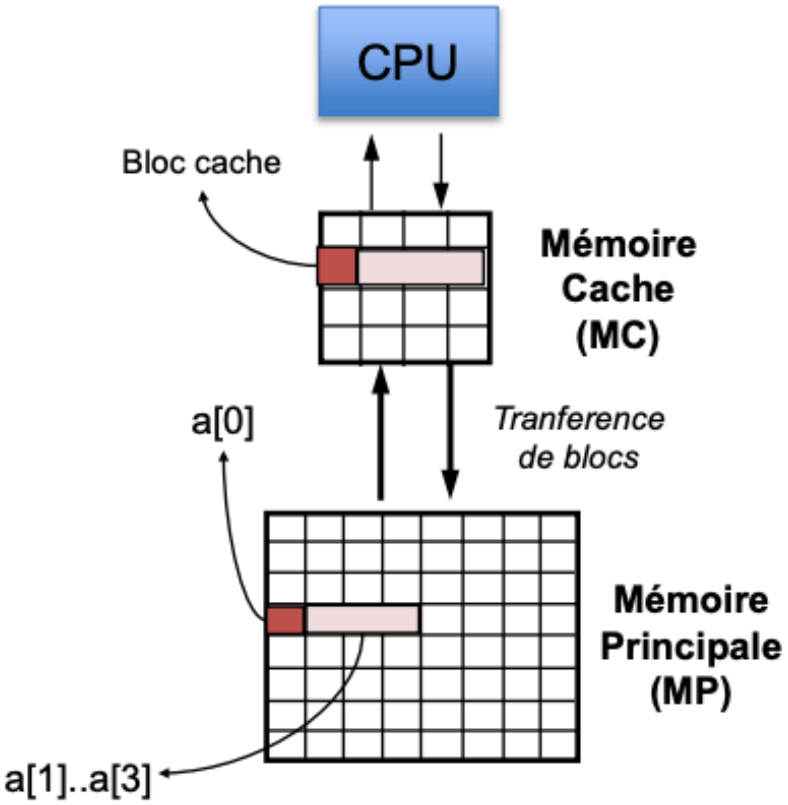
Chaque fois que le proceseur prend une donnée dans la mémoire principale (RAM), il va la copier dans la mémoire cache.

- Pénalisation dans le premier accès
- Accès suivants plus rapides

3.2. Localité spatiale

Lorsqu'une donnée est copiée dans la mémoire cache, elle est copiée avec les données qui l'entourent.

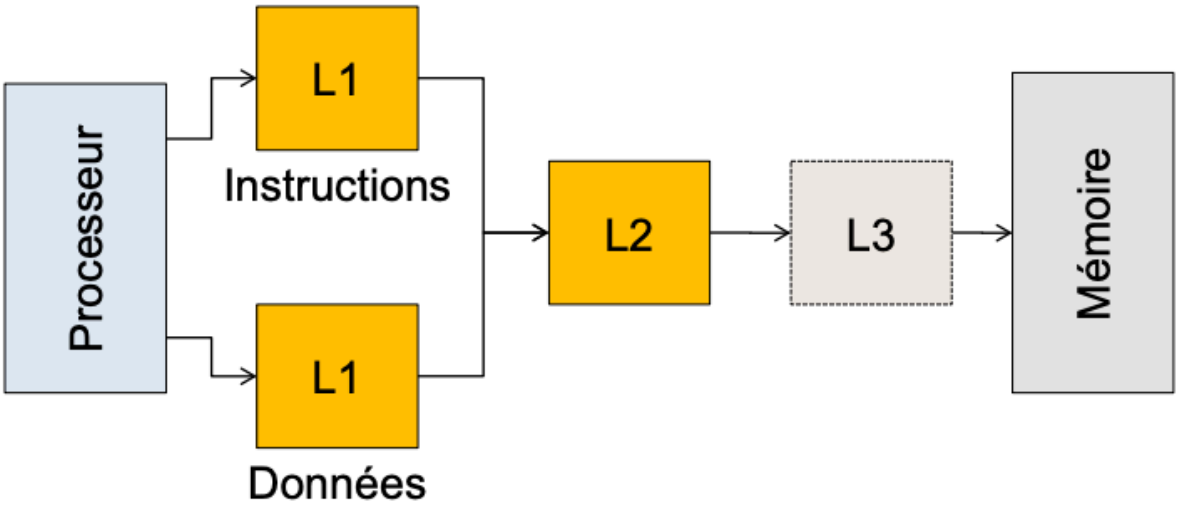
- Si une donnée est utilisée, les données qui l'entourent sont probablement utilisées aussi
- Accès à a[0]
- On prend aussi a[1], a[2], a[3]



4. Structure de la mémoire cache

4.1. Niveaux de cache

La mémoire cache est donc composée de plusieurs niveaux. De plus il existe à la fois une mémoire cache pour les données et une mémoire cache pour les instructions.



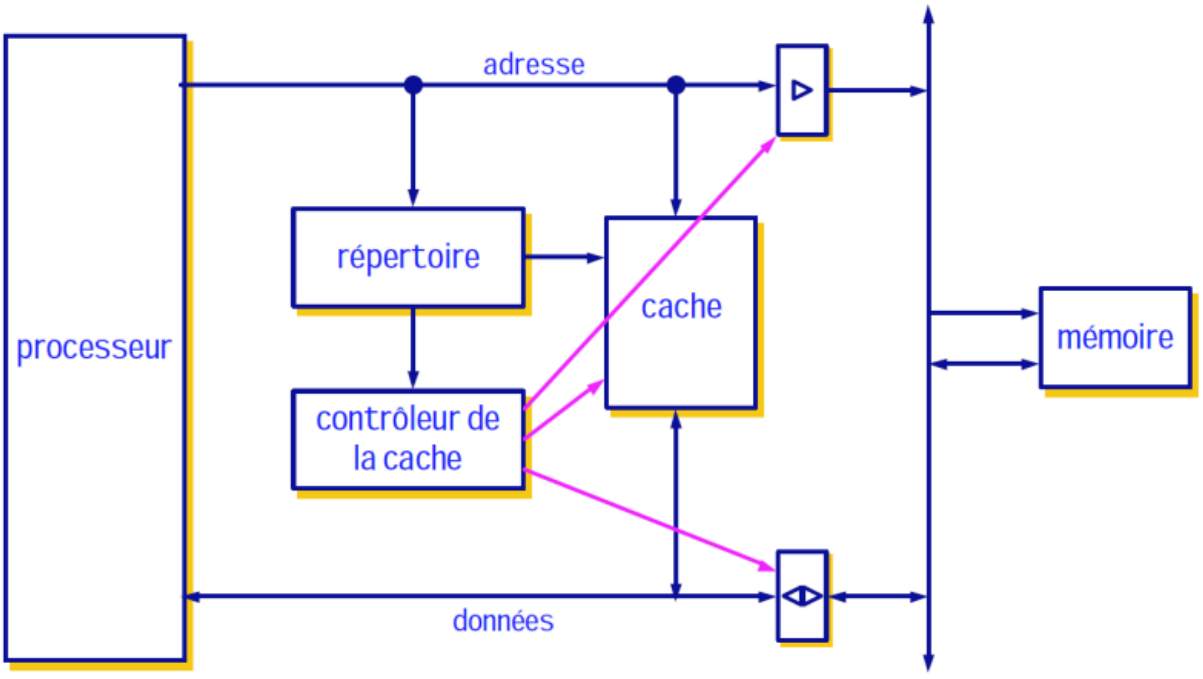
4.2. Organisation

Répertoire

- Indique si le mot accedé par le processeur se trouve dans la cache. Si oui, indique l'addresse de la cache où se trouve le mot.

Contrôleur de la cache

- Controle de la lecture/écriture en mémoire (parmi des buffers)

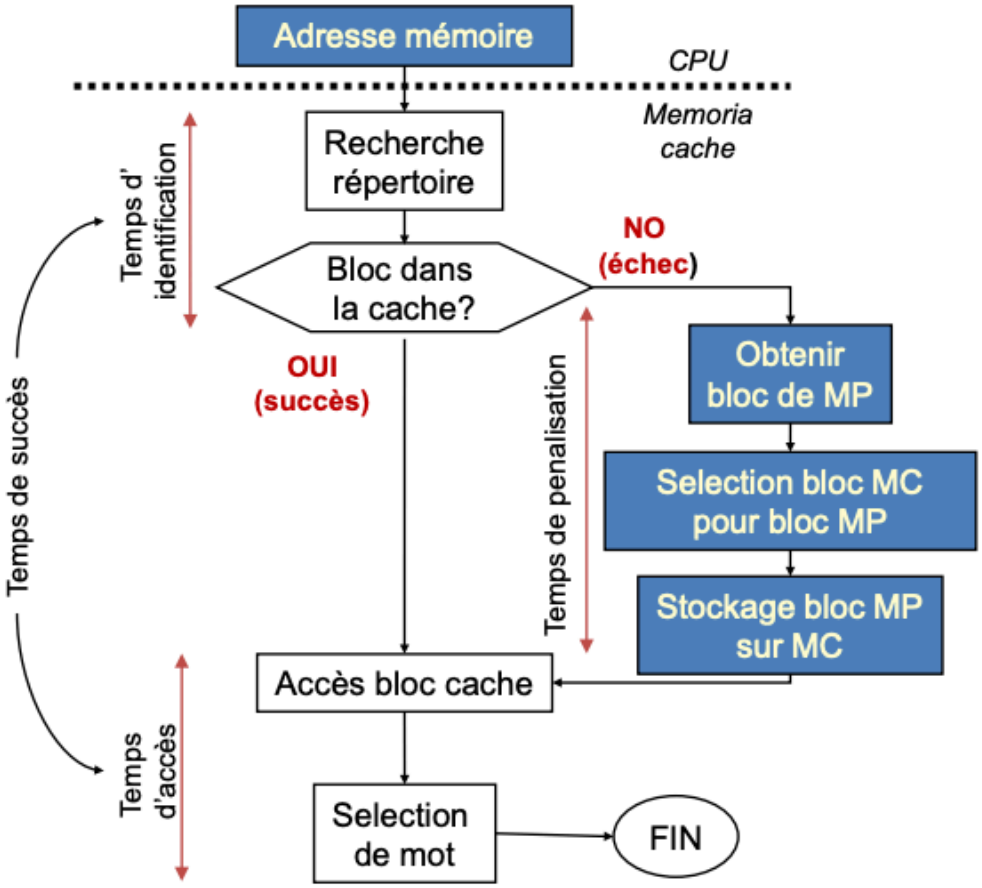


4.3. Structure interne

- La mémoire cache est composée de blocs/lignes de cache qui contiennent plusieurs mots de données. Un bloc/ligne est un **grpoupe d'octets (8-64)**
- Pour les lignes de 64 octets, les octets d'adresse 0-63 seront dans la même ligne, ceux de 64-127 dans une autres,…
- Le bloc/ligne a une **étiquette unique (tag)**

Les transferts entre cache et mémoire principale se font par lignes

4.4. Diagramme de flux



4.4.1. Objectifs

- Taux de succès maximum
- Temps d'accès minimum
- Temps penalisation minimum
- Reduir coût hardware

4.4.2. Calcul temps moyenne d'accès

Temps moyenne d'accès a mem = T_{MAM}

$T_{\text{MAM}} = T_{\text{succes}} + (1 - H) * T_{\text{penal}}$

$H =$ fréquence de succès

4.5. Fréquence de succès

- Dépend de la taille de la cache et des politiques de placement
- Miss penalty >> Hit time

	Rate	**Time**
Cache Hit	Hit rate =Nombre de hits/Nombre d'accès	Temps d'accès à la cache (hit time)
Cache Miss	1 - Hit rate	Temps d'accès à la mémoire principale + temps de chargement cache (miss penalty)

4.6. Recherche dans le cache

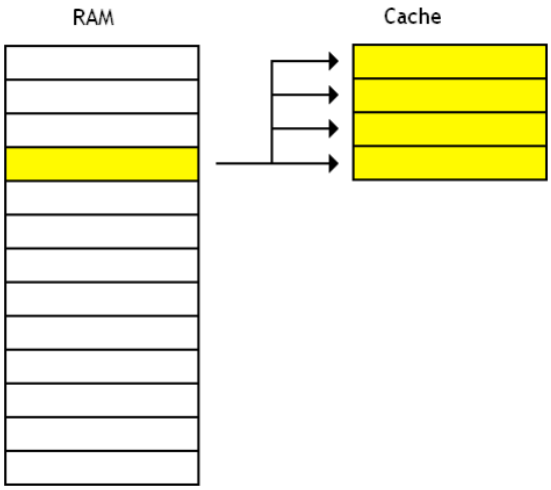
Consiste à trouver quelle est la ligne de cache dont le tag correspond à l'adresse de base demandée au répertoire.

3 stratégies (cablées, non logicielles) :

1. Fully Associative – Complètement associative
2. Direct Mapped – Associative par ensembles
3. Set Associative – Associative par voies

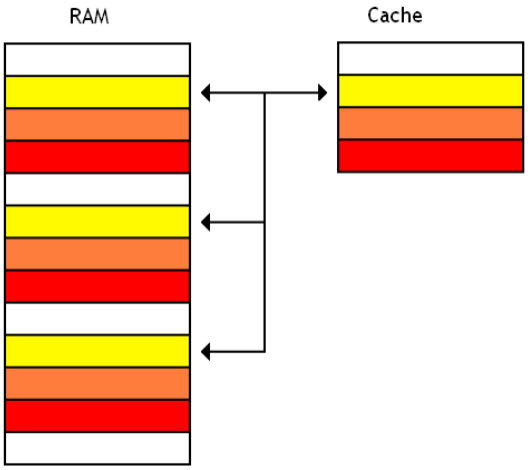
4.6.1. Fully Associative

- un mot de la mémoire principale peut être stocké n'importe où dans la cache
- Avantages : taux de succès très élevé (pas de conflit)
- Désavantages : trop lent (séquentiel, toutes les lignes à regarder)



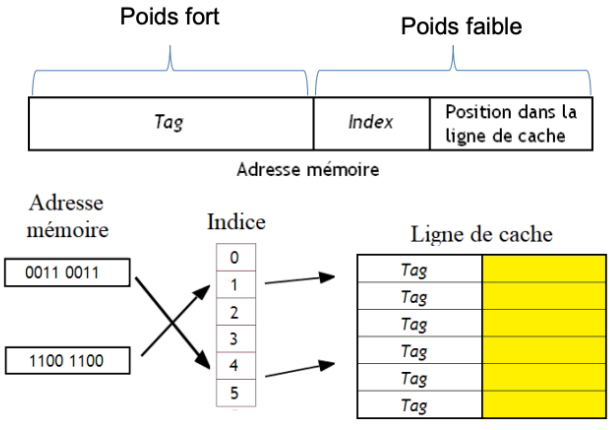
4.6.2. Direct Mapped

- un mot de la mémoire principale est chargé dans une ligne de cache prédéfinie (toujours la même)
- Avantages : accès rapide, car il faut vérifier qu'une ligne
- Désavantages : défaut par conflit, taux de succès mauvais



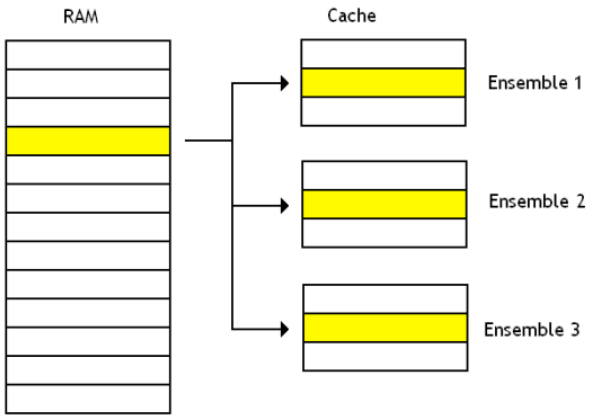
L'adresse physique (32 bits) est divisée en deux parties :

- Bits de poids faible permettent de spécifier un index, qui indique à quelle ligne de la cache l'information se trouve, ainsi que la position dans la ligne
- Bits de poids fort forment un tag, qui est à comparer avec la valeur stockée dans le répertoire
  - Bit de validité inclus



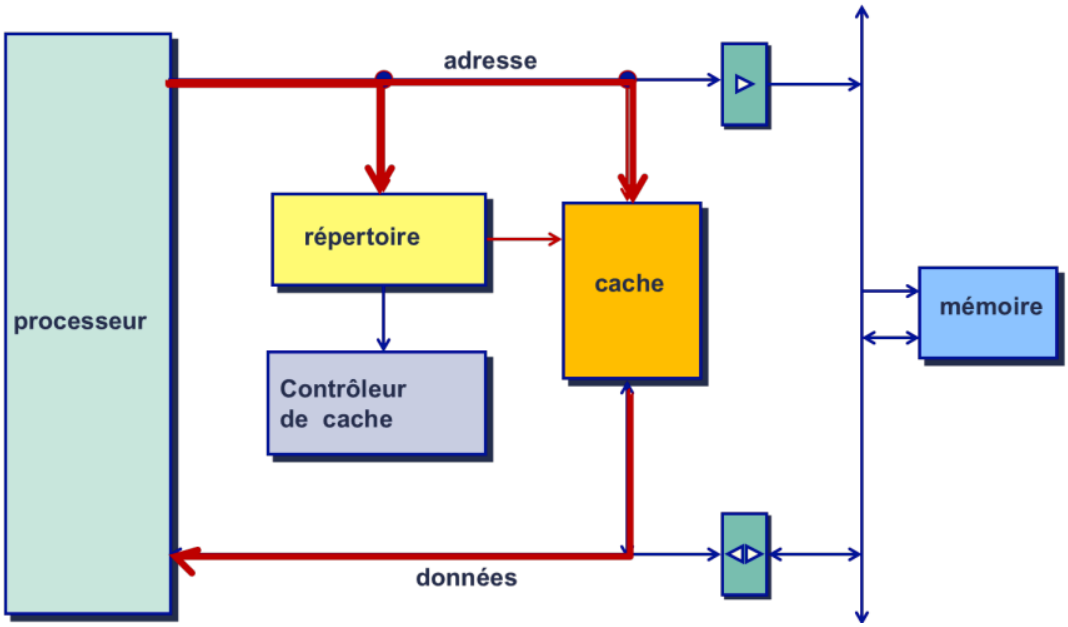
4.6.3. Set Associative

- Caches composés de plusieurs «caches» directement adressés accessibles en parallèle
- Chaque cache est appelé une voie
- Un mot de la mémoire peut être stocké en N positions différentes de la cache
- Diminution de la fréquence d'échec:
  - Associativité $*$ 2 = diminution de 20%
  - Taille cache $*$ 2 = diminution de 69%

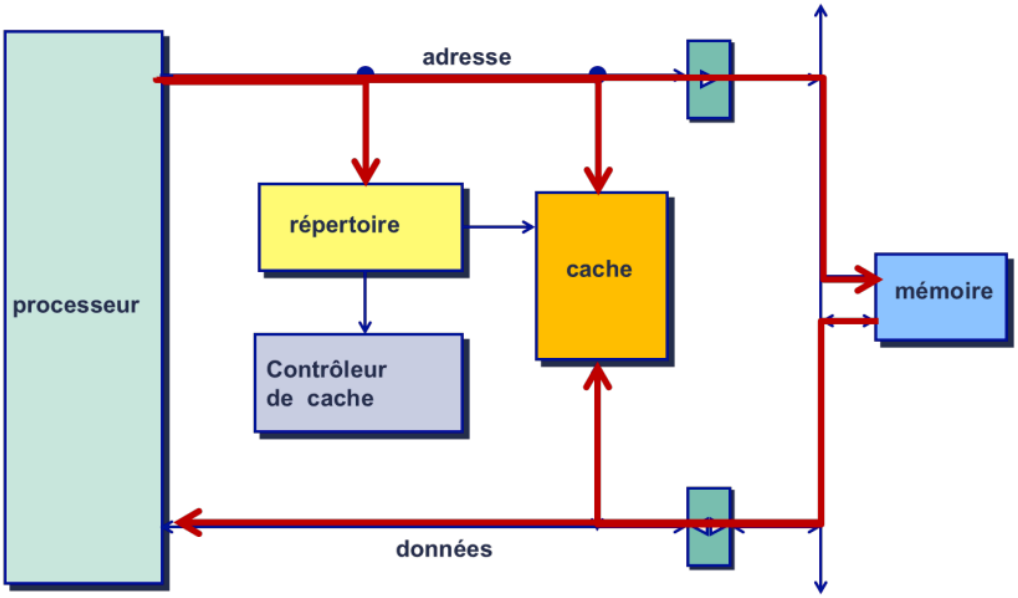


5. Hit & Miss

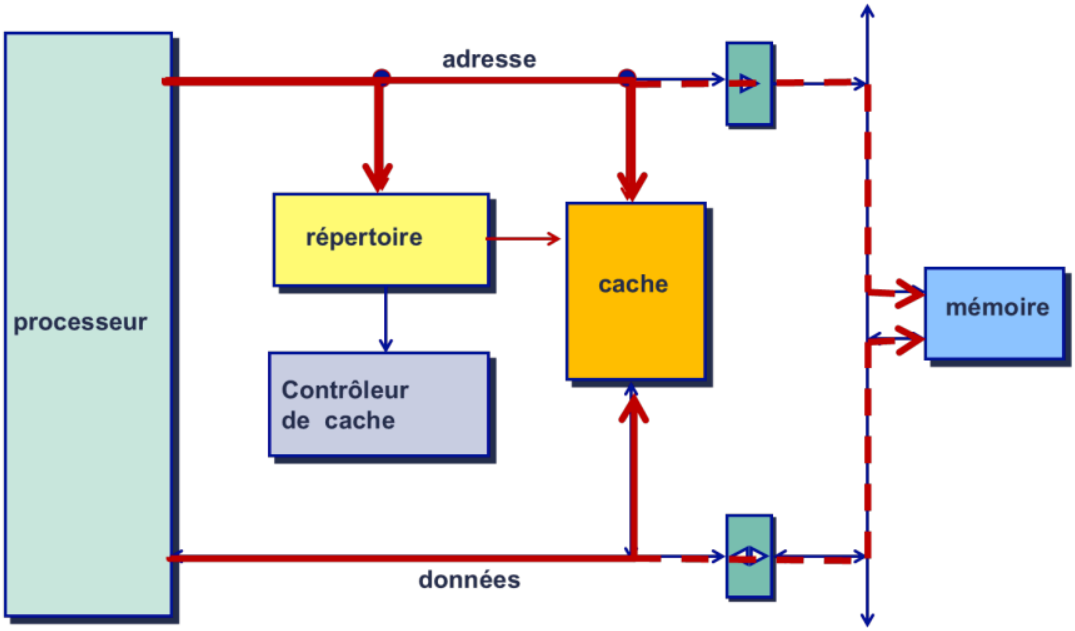
5.1. Hit lecture



5.2. Miss lecture



5.3. Hit écriture



5.4. Miss écriture