

INF 227

Caches ou anté-mémoires

B. Dupouy

Plan

- Hiérarchie des mémoires
- Généralités sur le fonctionnement des caches
- Organisation
- Condition de bon fonctionnement : principe de localité

Mémoires caches

Mémoires caches

Hierarchie de mémoires

- Du plus rapide au plus volumineux :

Cache matériel primaire (dizaine de kilo octets)
Cache matériel secondaire (Kilo ou méga octets)
Mémoire RAM (Méga octets)
Disque (Giga octets)

Mémoires caches

Introduit par WILKES (1965) sous le nom de mémoire esclave (*slave memory*), l'*antemémoire* ou *mémoire cache*, ou encore *cache*, est une mémoire intermédiaire située entre le processeur et la mémoire centrale et dont le temps d'accès est de 4 à 20 fois inférieur à celui de la mémoire centrale.

L'une des premières réalisations fut mise en œuvre sur les machines IBM 360/85 au cours des années 60 avec un cache de 16 kilo-octets. Lorsque l'unité centrale tente d'accéder à une information, en lecture ou en écriture, elle va d'abord la rechercher dans le cache. Si elle ne l'y trouve pas, elle va la prendre en mémoire centrale et recopie éventuellement dans le cache un bloc de mémoire centrale contenant cette information.

Les informations qui sont rangées dans le cache ne sont pas accessibles par programmation. Il est cependant possible d'inhiber son fonctionnement.

Ceci est indispensable lorsqu'on a besoin de mettre au point certains programmes de bas niveau, ou le système d'exploitation lui-même !

Rappel sur le principe de localité :

lorsqu'on accède à une information en mémoire (donnée ou instruction), il y a une forte probabilité pour que l'information suivante se trouve dans un voisinage de celle-ci.

Si l'on charge une portion de la mémoire dans le cache, il y a ainsi de grandes chances pour que le processeur n'ait à faire, pendant un certain temps, que des accès dans le cache, ce temps est évidemment fonction de la taille du cache et de la **structure des programmes**.

Mémoires caches

Généralités : *hit et miss*

- Soient h la probabilité pour qu'une information se trouve dans le cache (*hit ratio*), T_{cache} et $T_{\text{mémoire}}$ les temps d'accès respectifs au cache et à la mémoire, alors le temps de cycle moyen noté T_{eff} est :

$$T_{\text{eff}} = h T_{\text{cache}} + (1-h) T_{\text{mémoire}}$$

où $(1-h)$ est appelé *miss ratio*,

Si $\frac{T_{\text{mémoire}}}{T_{\text{cache}}} = 10$, une baisse de 1% sur un *hit ratio* voisin de 1 se traduit par une augmentation $\approx 10\%$ du temps d'accès moyen.

Mémoires caches

Un architecture de type *Harvard* pour les caches veut dire que l'on utilise deux caches: l'un pour les instructions, l'autre pour les données. En effet l'efficacité d'un cache repose sur le principe de localité et la notion de voisinage s'applique différemment pour les instructions et les données.

Mémoires caches

Généralités (2)

Données et instructions

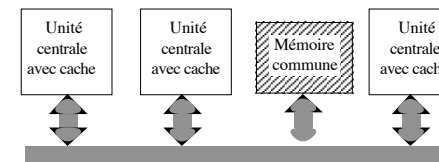
- Caches données et instructions séparés

Niveaux de caches

- Différents niveaux de cache : primaire et secondaire

Accès bus

- Le bus est libéré pendant les échanges entre processeur et cache. Cela permet une meilleure utilisation du bus principal si plusieurs processeurs y sont rattachés.



Mémoires caches

La politique qui consiste à écrire en mémoire immédiatement après l'écriture dans le cache est appelée *écriture immédiate* (*Write through*).

Dans le cas de l'écriture différée (*Write back*), on n'écrit en mémoire qu'au moment où l'on ne trouve plus l'information recherchée dans le cache : tant que les accès sont des *hits*, on n'écrit que dans le cache.

Cette méthode est plus performante que la première. Dans le cas de l'incrémentement du compteur d'une boucle, il est plus efficace de ne faire cette opération que dans le cache et non, à chaque itération, dans le cache *et* dans la mémoire. Par contre, cette méthode offre un inconvénient majeur en cas d'incident tel qu'une coupure de secteur.

Mémoires caches

Accès en lecture et en écriture

•lecture (d'une instruction ou d'une donnée) :

- si l'information se trouve dans le cache, accès en un temps réduit,
- sinon charger dans le cache un *bloc* de mots consécutifs, auquel appartient l'information cherchée.

Ce bloc sera désigné par le terme de *ligne* (*line*).

•écriture :

- si on fait référence à une information qui ne se trouve pas dans le cache, on met à jour la mémoire principale. On ne charge généralement pas la ligne dans le cache,
- si on trouve l'information dans le cache : write-thru ou write back

Mémoires caches

Une mémoire associative est constituée de deux éléments de mémorisation : l'un que nous appelons par *mémoire de contenu* (ou répertoire) et l'autre *mémoire d'adresses*.

On résoud avec ce type de mémoire le problème de l'accès rapide à une information dont on ne connaît pas l'adresse.

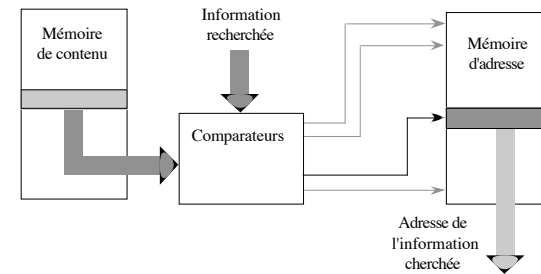
On range les informations elles-mêmes (ou une clé) dans la partie "mémoire de contenu" et leurs adresses dans la "mémoire d'adresses", on fait ensuite une comparaison **simultanée** entre l'information recherchée et celles qui sont dans la mémoire de contenu pour obtenir directement l'adresse cherchée.

On peut comparer la mémoire associative au glossaire d'un ouvrage qui permet d'obtenir rapidement l'adresse d'une information (son numéro de page), ou à un répertoire téléphonique qui donne l'adresse téléphonique d'une personne en fonction de son nom.

Mémoires caches

Mémoire associative

• Principe de fonctionnement



Mémoires caches

L'adresse émise par l'unité centrale est divisée en deux parties :

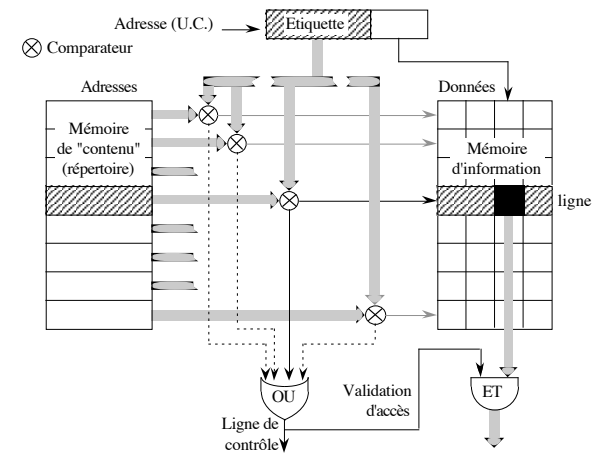
- en poids fort, une partie dite "étiquette" (*tag*) ,
- en poids faible, l'adresse de l'information dans la ligne

Lors d'un accès à la mémoire, l'étiquette est comparée à l'adresse rangée dans la partie "contenu" (ou répertoire) de la mémoire associative. S'il y a égalité le processeur accède directement à l'information qui se trouve dans le cache.

Mémoires caches

Etiquette et ligne (tag, line)

- L'étiquette est comparée à l'information rangée dans la partie "contenu" (ou répertoire) de la mémoire associative. S'il y a égalité on accède directement à l'information qui se trouve dans la ligne du cache.



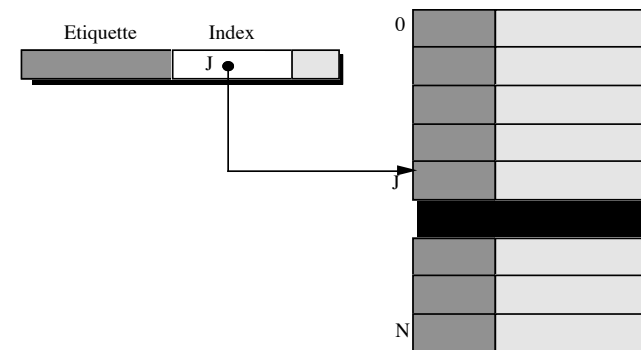
Mémoires caches

Contrairement au cache associatif, on ne dispose plus que d'un seul comparateur. Par contre, le décodage de l'index introduit un retard supplémentaire.

Mémoires caches

Cache à accès direct (direct access)

- Objectif : imiter le nombre de comparateurs.
- L'adresse de l'information à chercher dans le cache est divisée en une partie **index** et une partie **étiquette**.
- La partie **index** sélectionne une entrée du cache, mais cette entrée est partagée par toutes les adresses ayant le même index,
- La partie étiquette permet de vérifier si cette entrée concerne bien l'adresse courante.



Mémoires caches

microprocesseur MOTOROLA® 68020

Les adresses émises sont des adresses d'octets. L'index, sur 6 bits, permet d'adresser la mémoire de contenu disposant de $2^6 = 64$ entrées. La mémoire "information" permet de stocker 64 mots de 32 bits, soit 256 octets.

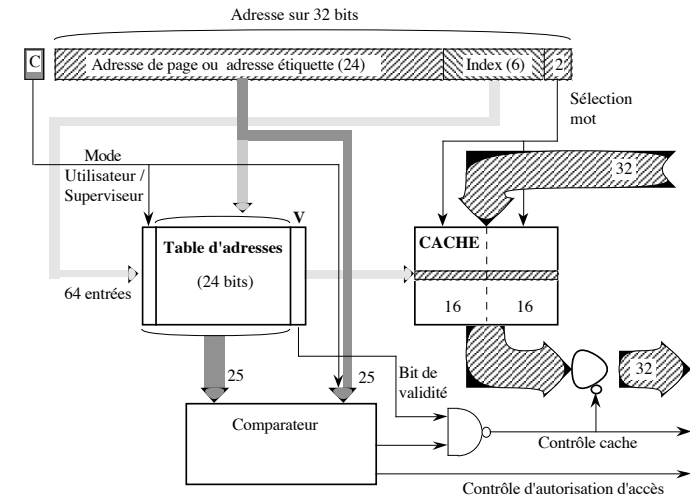
Si cours d'une opération de lecture, l'information recherchée n'est pas trouvée dans le cache, il y a transfert d'un bloc de quatre octets de la mémoire vers le cache.

On pourra noter, sur le schéma, que la comparaison s'effectue sur 25 bits et non sur 24. En effet, certains accès autorisés en mode *superviseur* ne le sont plus en mode *utilisateur*. Le bit C, indicatif du mode dans lequel se trouve le processeur, est pris en compte par le comparateur pour déclencher un éventuel déroutement consécutif à un accès non autorisé.

Mémoires caches

Exemple de cache à accès direct

- Le schéma suivant s'inspire du MOTOROLA® 68020. (index sur 6 bits), ligne de quatre octets :



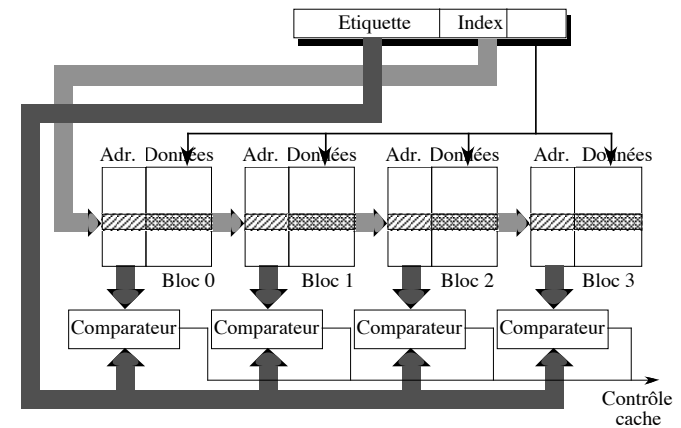
Mémoires caches

Cette réalisation conjugue les deux méthodes décrites précédemment. Elle revient à utiliser plusieurs caches à accès direct montés en parallèle. Elle est associative dans le sens où les informations adressées dans tous les blocs sont comparées simultanément à la partie "étiquette" de l'adresse.

Mémoires caches

Cache à accès associatif par blocs (set associative)

- Les entrées désignées par l'index dans tous les blocs sont comparées simultanément à la partie "étiquette" de l'adresse.



- le nombre de bloc est désigné par *set associativity*

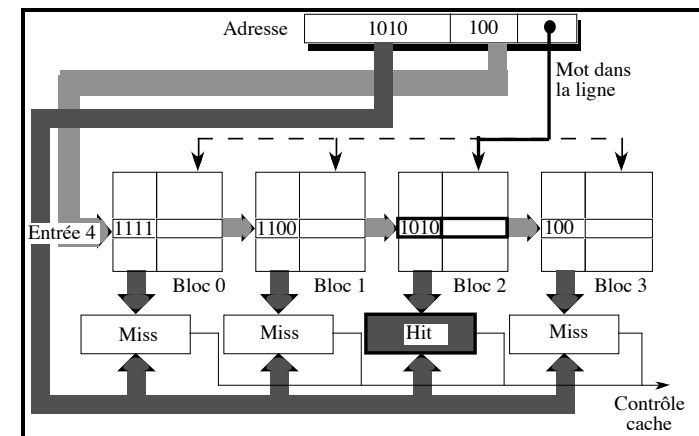
Mémoires caches

Mémoires caches

Accès en mode "set associative"

- Exemple d'accès au cache précédent (on suppose que la ligne fait 16 octets) avec l' adresse suivante, codée en binaire :

1010100xxxx



Mémoires caches

Mémoires caches

Quelques définitions ...

Associativity (A)	Nombre de places où peut se trouver une ligne
Capacity (C)	Nombre d'octets contenus dans le cache
Capacity miss	<i>miss</i> dans un cache purement associatif géré LRU
Compulsory miss	<i>miss</i> sur la première référence
Conflict miss	<i>miss</i> dans un cache <i>A-ways</i> qui serait un <i>hit</i> dans un cache purement associatif
Miss penalty	temps de chargement du cache lors d'un <i>miss</i>
Purement associatif (fully associative)	une ligne de taille L peut être placée n'importe où (ici, $A = C/L$)
Set associative	un cache dans lequel une ligne peut se trouver à A places différentes.

