Section : 2ème Année Ingénieur d'Etat Informatique

Chapitre 1

Les Mémoires

Chargée du cours : Pr. Ch. Bencheriet

Année Universitaire : 2024/2025

Plan du cours

1. Introduction

- Dans un ordinateur toutes les informations : valeur numérique, instruction, adresse, symbole (chiffre, lettre,…)...etc, sont manipulées sous une forme binaire.
- Ces informations doivent en général être conservées pendant un certain temps pour permettre leur exploitation.
- Ce rôle est dévolu aux mémoires chargées de conserver programmes, données provenant de l'extérieur, résultats intermédiaires, données à transférer à l'extérieur, …etc.

2. Qu'est ce qu'une mémoire ?

Mémoire = tout dispositif capable de stocker des informations (instructions et données) de telle sorte que l'organe qui les utilise puisse à n'importe quel moment accéder à l'information qu'il demande.

- Les informations peuvent être écrites ou lues.
- Il y a **écriture** lorsqu'on enregistre des données en mémoire,
- Il y a **lecture** lorsqu'on récupère des informations précédemment enregistrées.
- La lecture peut être destructive (l'information lue n'est plus en mémoire) ou non.

3. Types de mémoires?

4. Notion de hiérarchie mémoire

Un ordinateur est composé de plusieurs types de mémoire. On peut d'abord distinguer la mémoire principale à l'interne et les mémoires périphériques à l'externe (appelées aussi mémoires auxiliaires ou mémoires de masse ou alors secondaire).

Les principaux critères à retenir pour le choix d'une mémoire sont la capacité, la vitesse et le coût.

- Une mémoire idéale serait une mémoire de grande capacité, possédant un temps d'accès très faible.
- Les mémoires de grande capacité sont souvent très lentes et que les mémoires rapides sont très chères.
- On utilise des mémoires de faible capacité mais très rapide pour stocker les informations dont le microprocesseur se sert le plus et on utilise des mémoires de capacité importante mais beaucoup plus lente pour stocker les informations dont le microprocesseur se sert le moins.
- Ainsi, plus on s'éloigne du microprocesseur et plus la capacité et le temps d'accès des mémoires vont augmenter.

4. Notion de hiérarchie mémoire

1. **Les registres:** éléments de mémoire les plus rapides. situés au niveau du processeur et servent au stockage des opérandes et des résultats intermédiaires.
2. **La mémoire cache:** mémoire rapide de faible capacité destinée à accélérer l'accès à la mémoire centrale en stockant les données les plus utilisées.
3. **La mémoire principale:** organe principal de rangement des informations. contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.
4. **La mémoire d'appui:** sert de mémoire intermédiaire entre la mémoire centrale et les mémoires de masse. Elle joue le même rôle que la mémoire cache.
5. **La mémoire de masse:** mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations. EX: supports magnétiques (disque dur, disquette) ou optiques (CDROM, DVDROM).

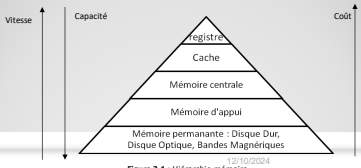


Figure 2.1 : Hiérarchie mémoire

5. Caractéristiques d'une mémoire

- **La capacité** / Capacity
- **Le format des données** / Data format
- **Le temps d'accès** / Access time
- **Le débit** / Throughput
- **Volatilité** / Volatility

5. Caractéristiques d'une mémoire

1. **La capacité :** le nombre total de bits que contient la mémoire. s'exprime en octet.

En informatique, les capacités mémoires sont en général des multiples de puissances de 2.

Pour cette raison, les informaticiens de la première heure avaient l'habitude d'utiliser les préfixes : *kilo*, *méga*, giga… etc. comme des puissances de 2.

Toutefois la normalisation des préfixes binaires de **1998** par la **Commission électrotechnique internationale** spécifie les préfixes suivants pour représenter les puissances de 2 :

- **kibi** pour « **ki**lo **bi**naire » ;
- **mébi** pour « **méga bi**naire » ;
- **gibi** pour « **gi**ga **bi**naire » ;
- **tébi** pour « **té**ra **bi**naire » ;

5. Caractéristiques d'une mémoire

Il est donc préférable d'utiliser ces préfixes (kibi = Ki, mébi = Mi, gibi = Gi,… etc.), et de laisser aux préfixes **SI** (Système International d'unités) leur sens recommandé (kilo = k , méga = M , giga = G,… etc.).

Nom	Préfixe	Symbole	Capacité
kilo binaire	kibioctet	Kio	2^{10}_ 1024 octets
méga binaire	mébioctet	Mio	2^{20} = 1048576 octets
giga binaire	gibioctet	Gio	2^{30} = 1 073 741 824 octets
téra binaire	tébioctet	Tio	2^{40} = 1 099 511 627 776 octets
péta binaire	pébioctet	Pio	2^{50} = 1 125 899 906 842 624 octets
exa binaire	exbioctet	Eio	2^{60} = 1 152 921 504 606 846 976 octets
zetta binaire	zébioctet	Zio	2^{70} = 1 180 591 620 717 411 303 424 octets
yotta binaire	yobioctet	Yio	2^{80} = 1 208 925 819 614 629 174 706 176 octets

5. Caractéristiques d'une mémoire

2. **Le format des données :** le nombre de bits que l'on peut mémoriser par case mémoire. On dit aussi que c'est la largeur du mot mémorisable.

3. **Le temps d'accès :** le temps qui s'écoule entre l'instant où a été lancée une opération de lecture/écriture en mémoire et l'instant où la première information est disponible sur le bus de données.

4. **Le débit :** le nombre maximum d'informations lues ou écrites par seconde.

$$Débit\ binaire = \frac{Longeur\ du\ mot}{Longueur\ du\ cycle}$$

Exemple : Une mémoire ayant un cycle de 1.2 µs et des mots de 36 bits. Le taux de transfert sera: 36 bits / 1.2 x10 $^{-6}$ sec = 30 x 10^{6} bps (bit par seconde)

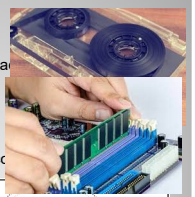
5. **Volatilité:** caractérise la permanence des informations dans la mémoire. L'information stockée est volatile si elle risque d'être altérée par un défaut d'alimentation électrique et non volatile dans le cas contraire.

Méthodes d'accès à la mémoire

Accès séquentiel

Pour accéder à une information on doit parcourir toutes les informations précédentes . Accès lent

Exemple : bandes magnétiques (K7 vidéo)



Accès direct

Chaque information a une adresse propre. On peut a… chaque adresse

Exemple : mémoire centrale

Accès semi-séquentiel

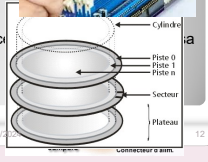
Intermédiaire entre séquentiel et direct.

Exemple : disque dur : Accès direct au cylindre , Acc… sur un cylindre

Accès associatif/par le contenu

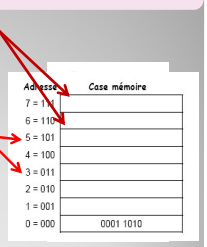
Une information est identifiée par une clé. On acc… clé

Exemple : mémoire cache



6. Organisation d'une mémoire

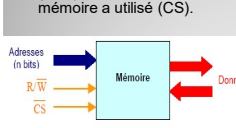
- Une mémoire peut être représentée comme une armoire de rangement constituée de différents tiroirs.
- Chaque tiroir représente une case mémoire: mot mémoire
- Le nombre de cases mémoires est identifier par un numéro appelé adresse.
- Chaque donnée devient accessible grâce à son adresse
- Avec une adresse de ***n*** bits il est possible de référencer 2^n cases mémoire.
- Chaque case est remplie par un mot de données (sa longueur *m* est toujours une puissance de 2)



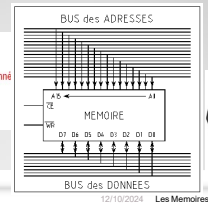
Organisation d'une mémoire

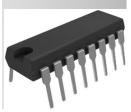
6. Organisation d'une mémoire a semi-conducteur

- Nombre fils d'adresses d'un boîtier mémoire définit le nombre cases mémoire que comprend le boîtier. **comment?**
- Nombre fils de données définit la taille des données que l'on peut sauvegarder dans chaque case mémoire.
- Un boîtier mémoire comprend aussi une entrée de commande qui permet de définir le type d'action effectuées avec la mémoire (lecture/écriture) (R/W) et une entrée de sélection qui permet de sélectionner le boîtier mémoire a utilisé (CS).

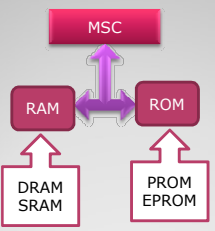


Présentation d'un circuit mémoire





7. Technologies de construction des Mémoire a semi-conducteur

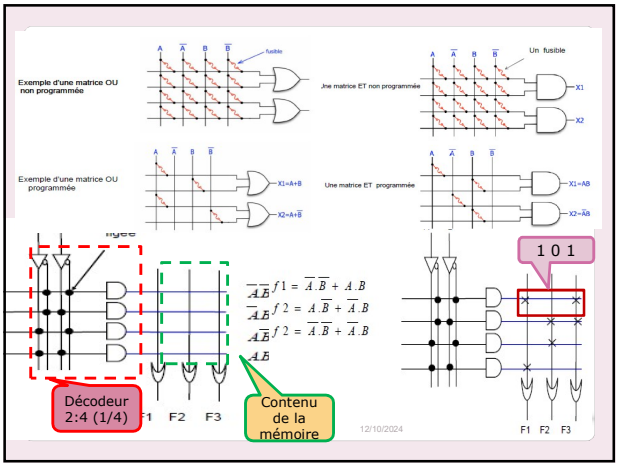


7.2 Mémoires Mortes

- Dans certaines applications, il est nécessaire de pouvoir conserver des informations (programmes, données, …) de façon permanente même lorsque l'alimentation électrique est interrompue.
- On utilise alors des mémoires mortes ou mémoires à lecture seule (ROM : Read Only Memory).
- Dans ce cas les informations contenues en mémoire ne peuvent être accédées qu'en lecture.
- Les informations ne peuven… accidentellement. Dans cette c…
  1. PROM
  2. EPROM
     a. EEPROM
     b. UV EPROM
     c. FLASH.

OTP ROM (one-time programmable) ne peut être programmée qu'une fois à l'aide d'un … (**E**lectrically-**E**rasable **PROM**) : mémoire morte … le contenu effacé par ultra-violets (plusieurs …

le contenu est efface électriquement et plus rapidement que sur les EEPROM



7.2 Mémoires Mortes

Domaines d'applications des ROMs

- La conversion de code,
- La génération de caractères pour un terminal ou une imprimante,
- Les ROMs peuvent être également utilisées pour conserver certains programmes et les données associées (les calculateurs de poche ou pour les programmes de démarrage et de chargement qui s'exécutent automatiquement lors de la mise sous tension d'un ordinateur).

7.1 Technologie des mémoires vives

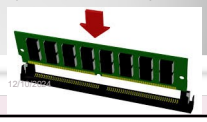
On peut réaliser des RAM avec deux technologies différentes : les **RAM** **D**ynamiques **(DRAM)**, et les **RAM S**tatiques **(SRAM)**

RAM dynamiques (DRAM) :

- Mémoire très utilisé car peu coûteuse.
- Risque de perdre son contenu s'il n y' a pas un rafraîchissement périodique de ses cellules (environ 1000 fois par seconde) par un circuit de rafraîchissement qui lit l'état de chaque cellule et le réécrit,
- cette perte d'information et le rafraîchissement constant qui fait de la RAM cette appellation (dynamique).
- On trouve des boîtiers DRAM de 256k x 1 bit, 256k x 4 bits, 1 M x 1 bit, 18 M x 4 bits.

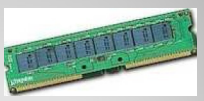
RAM statiques (SRAM) :

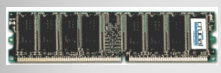
- Appeler statique car tant que la mémoire est alimenter électriquement elle maintien l'information enregistrer dans ces cellules (0 ou 1).
- Le temps d'accès de ces mémoires est plus court que celui des DRAM.
- Elles sont plus coûteuses (demande 4 fois plus de transistors que les DRAM).



7.1 Technologie des mémoires vives

Exemple : **FPM** (Fast Page Mode), **EDO** (Extended Data Output), **SDRAM** (Synchronous), **DDRRAM** (Double Data Rate), **DRRAM** (Direct Rambus), **SLRAM** (Sync Link).





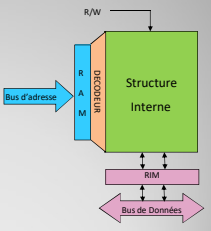
Remarque

En général les mémoires dynamiques, qui offrent une plus grande densité d'information et un coût par bit plus faible, sont utilisées pour la mémoire centrale.

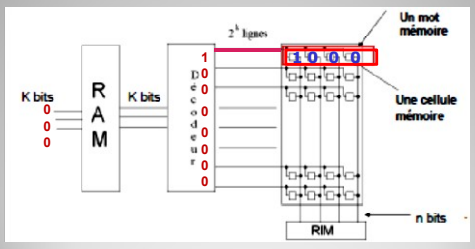
Les mémoires statiques, plus rapides, sont utilisées lorsque le facteur vitesse est critique, notamment pour des mémoires de petite taille comme les caches et les registres.

8. Structure physique d'une mémoire

- Registre d'Adresse Mémoire **(RAM)**
- Registre d'information mémoire **(RIM)**
- **Décodeur** : permet de sélectionner un mot mémoire.
- **R/W** : commande de lecture/écriture
- Bus d'adresses de taille k bits
- Bus de données de taille n bits



8.1 Sélection d'un mot mémoire



8.2 Calcul de la capacité d'une Mémoire

Soit :

k la taille du bus d'adresses (taille du registre RAM),

n la taille du bus de données (taille du registre RIM ou la taille d'un mot mémoire)

La capacité de la mémoire centrale est exprimée en nombre de mots mémoire ou en bits (octets, kilo-octets,….)

La capacité = 2^k Mots mémoire ou bien **La capacité = 2^k * n Bits**

Exemple

Dans une mémoire la taille du bus d'adresses K=14 et la taille du bus de données n=4. Calculer la capacité de cette mémoire ?

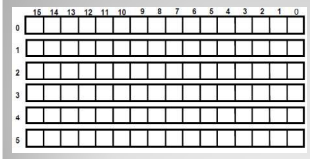
Solution

$C=2^{14}$ = 16384 Mots de 4 bits

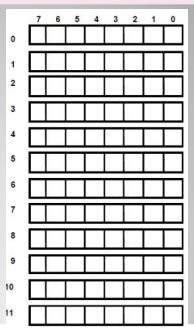
$C= 2^{14} * 4$ = 65536 Bits = 8192 Octets = 8 Kio

8.2 Calcul de la capacité d'une Mémoire

Il existe plusieurs façons d'organiser une mémoire. À titre d'exemple, une mémoire formée de 96 bits peut avoir les organisations suivantes :



96 bits organisés en 6 mots de 16 bits



96 bits organisés en 12 mots de 8 bits

8.3 Lecture et écriture d'une information en MC

Lecture

- Charger dans le registre RAM l'adresse du mot à lire.
- Lancer la commande de lecture (R/W=1)
- L'information est disponible dans le registre RIM au bout d'un certain temps (temps d'accès).

Ecriture

- Charger dans le RAM l'adresse du mot ou se fera l'écriture.
- Placer dans le RIM l'information à écrire.
- Lancer la commande d'écriture pour transférer le contenu du RIM dans la mémoire

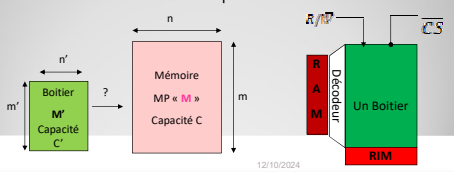
8.4 Conception des Mémoires

Un boîtier Mémoire possède la même structure qu'une mémoire (RAM, RIM,….) en plus de la commande $\overline{CS}$

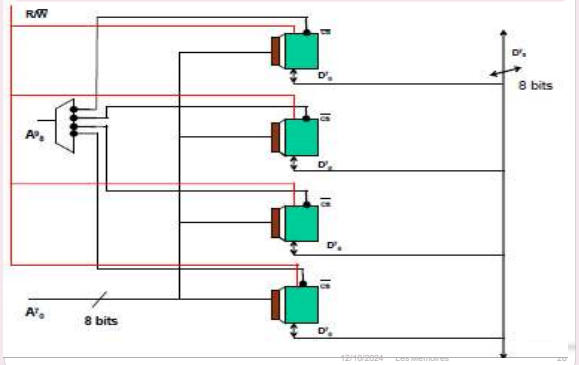
On suppose que C > C' (m >= m', n >=n')

Le nombre de boîtiers pour réaliser la MP « M » nécessite la connaissance des facteurs :

1. **P** : détermine le nombre de boîtiers M' nécessaire pour obtenir le nombre de mots de la mémoire M (extension lignes). **P = m/m'**
2. **Q** : détermine le nombre de boîtier M' nécessaire pour obtenir la taille de mot de la mémoire M (extension mots ou extension colonnes). **Q= n/n'**
3. Le nombre total de boîtiers nécessaire pour réaliser la mémoire M est : **M' = P.Q**

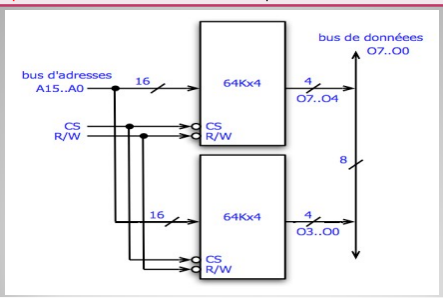


Extension de capacité des mémoires



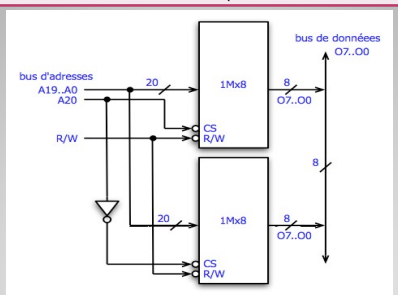
Extension de capacité des mémoires

Exemple 2: utilisation de deux RAM 64Kx4 pour obtenir une mémoire 64Kx8



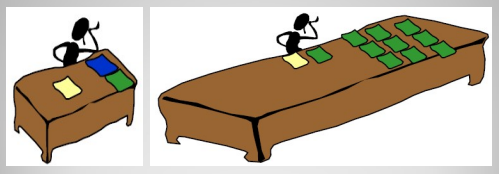
Extension de capacité des mémoires

Exemple 3: utilisation de deux RAM 1Mx8 pour obtenir une mémoire 2Mx8



Remarque (importance de la taille mémoire)

➢ Si sa mémoire de travail est petite, seul des petits programmes pourront être traités par le processeur → nécessité d'aller chercher les informations en mémoire secondaire d'où le ralentissement important de ses performances,
➢ Si la mémoire de travail est importante → les échanges avec la mémoire secondaire seront beaucoup moins nombreux et les performances du processeur seront meilleures.



Mémoires accès séquentiel (FIFO/LIFO)

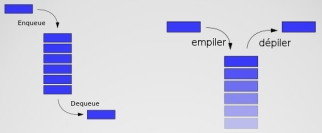
conservent les données triées dans l'ordre d'écriture (l'ordre d'arrivée)

Mémoire FIFO (First In First Out) ou File

- La file d'attente ou **FIFO** signifie premier entré premier sorti.
- Mettre en attente une donnée (enqueue, en anglais) et lire la donnée la plus ancienne (dequeue, en anglais).

Mémoire LIFO (Last In First Out) ou Pile

- L'abréviation **LIFO** signifie : dernier entré premier sorti.
- Les valeurs s'empilent les uns sur les autres (push) et le premier qui peut être récupérer c'est celui qui se trouve au sommet (pop).
- Les données ne peuvent être récupérer que dans l'ordre inverse de leur arrivées.



Mémoires accès séquentiel (FIFO/LIFO)

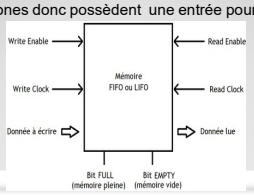
conservent les données triées dans l'ordre d'écriture (l'ordre d'arrivée)

Caractéristiques communes

- Aucun bus d'adresse vu que ces mémoires ne sont pas adressables.
- Deux sorties nommées **FULL** et **EMPTY.**

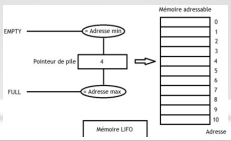
FULL (mémoire pleine) : ne peut plus accepter de nouvelle donnée en écriture.
EMPTY (mémoire Vide) : il n'y a pas possibilité de lire son contenu.

- Port de lecture et écriture souvent séparés (rarement le même).
- Mémoires synchrones donc possèdent une entrée pour le signal d'horloge.



Mémoire LIFO ou Pile

- Un seul pointeur géré par un compteur / décompteur
- Les mémoires LIFO peuvent se concevoir en utilisant une mémoire RAM, couplée à un registre.
- Les données y sont écrites à des adresses successives : on commence par remplir la RAM à l'adresse 0, puis on poursuit adresse après adresse, ce qui garantit que la donnée la plus récente soit au sommet de la pile.
- Tout ce qu'il y a à faire est de mémoriser l'adresse de la donnée la plus récente, dans un registre appelé le **pointeur de pile**.
- Cette adresse donne directement la position de la donnée au sommet de la pile, celle à lire lors d'une lecture.
- Le pointeur de pile est incrémenté à chaque écriture, pour pointer sur l'adresse de la nouvelle donnée. De même, il est décrémenté à chaque lecture, vu que les lectures sont destructrices (elles effacent la donnée lue).

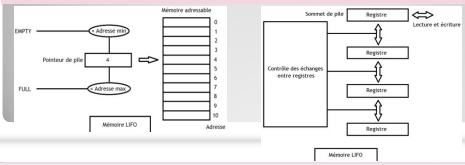


Mémoire LIFO ou Pile

- La gestion des bits EMPTY et FULL est relativement simple : il suffit de comparer le pointeur de pile avec l'adresse minimale et maximale.

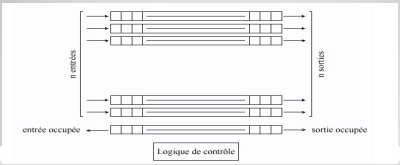
1. Si le pointeur de pile et l'adresse maximale sont égaux, cela signifie : la mémoire est **pleine**.
2. Quand le pointeur de pile pointe sur l'adresse minimale (0), la mémoire est **vide**.

NB: Il est aussi possible, de créer des LIFO à partir de registres. Pour cela, il suffit d'enchainer des registres les uns à la suite des autres. Les données peuvent passer d'un registre à son suivant, ou d'un registre aux précédents. Toutes les lectures ou écritures ont lieu dans le même registre, celui qui contient le sommet de la pile.



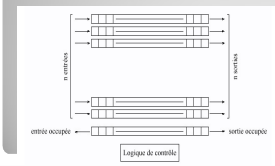
Mémoire FIFO ou File

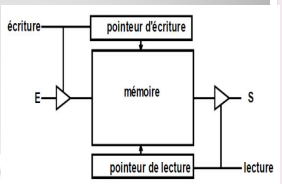
- L'accès n'est pas aléatoire,
- l'ordre en sortie est identique à celui en entrée.
- Il n'y a pas d'adressage.
- Une file est constituée de n registres à décalage comptant chacun m cases.
- Le nombre n correspond à la largeur des mots, alors que m est la profondeur de la file ou sa capacité.
- Une file dispose de deux bus de données distincts (double ports) en entrée et en sortie soit 2 pointeurs de lecture et d'écriture gérés par des compteurs et un plan mémoire RAM (SRAM ou DRAM)



Mémoire FIFO ou File

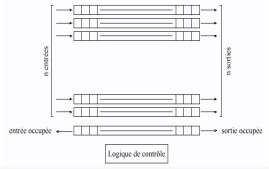
- Le bit R/W est souvent séparé en deux bits distincts : un bit Write Enable sur le port d'écriture, qui indique qu'une écriture est demandée, et un bit Read Enable sur le port de lecture qui indique qu'une lecture est demandée.
- La séparation du bit R/W en deux bits séparés pour chaque port est justifiée par:
  1. Elles ont (souvent) une fréquence pour la lecture qui est distincte de la fréquence pour l'écriture. En conséquence, elles reçoivent deux signaux d'horloge sur deux entrées séparées : un pour la lecture et un pour l'écriture.
  2. La présence de deux fréquences s'explique par le fait que les mémoires FIFO servent à interfacer deux composants qui ont des vitesses différentes, comme un disque dur et un processeur,

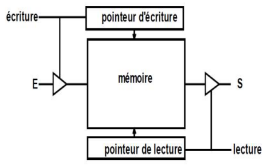




Mémoire FIFO ou File

- L'utilisateur peut écrire dans la FIFO si le premier étage est **libre** et lire la FIFO que si le dernier étage est **occupé**.
- Un registre interne indique l'état (**libre** ou **occupé**) de chacun des étages.
- Seules deux opérations sont possibles sur de telles mémoires : mettre en attente une donnée (enqueue, en anglais) et lire la donnée la plus ancienne (dequeue, en anglais).





Mémoire FIFO ou File

Où utilisent-on les FIFO?

- Utilisées pour mettre en attente des données ou commandes, tout en conservant leur ordre d'arrivée d'où l'appelation de **mémoire tampon**. *L'utilisation principale des mémoires tampons est l'interfaçage de deux composants de vitesse différentes qui doivent communiquer entre eux.*
- On retrouve des mémoires tampons dans beaucoup de matériel électronique : dans les disques durs, des les lecteurs de CD/DVD, dans les processeurs, dans les cartes réseau, etc.

Mémoire FIFO ou File

Où utilisent-on les FIFO?

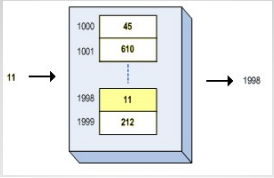
Exemples

Disque dur : reçoit régulièrement, de la part du processeur, des commandes de lecture écriture et les données associées. Mais le disque dur étant un périphérique assez lent, il doit mettre en attente les commandes/données réceptionnées avant de pouvoir les traiter. Et cette mise en attente doit conserver l'ordre d'arrivée des commandes, sans quoi on ne lirait pas les données demandés dans le bon ordre. Pour cela, les commandes sont stockées dans une mémoire FIFO et sont consommées au fur et à mesure.

Cartes réseau : reçoivent des paquets de données à un rythme discontinu, qu'elles doivent parfois mettre en attente tant que la carte réseau n'a pas terminé de gérer le paquet précédent.

Mémoires associatives

Les mémoires associatives ont un fonctionnement totalement opposé aux mémoires adressables, au lieu d'envoyer l'adresse pour accéder à la donnée, on envoie la donnée pour récupérer son adresse. On les appelle aussi des **mémoires adressables par contenu**, ou encore ***Content-adressed Memory*** (CAM).



Mémoires associatives

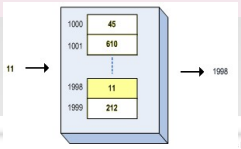
Pourquoi le processeur utilise ces mémoires?

- Généralement, les données manipulées par un programme sont regroupées dans des structures de données organisées : tableaux, listes, graphes, arbres, etc. Et il arrive fréquemment que l'on recherche une donnée bien précise dedans.
- Certaines structures de données sont conçues pour accélérer cette recherche, ce qui donne des gains de performance bienvenus, mais au prix d'une complexité de programmation non-négligeable.
- Les mémoires associatives sont une solution matérielle alternative bien plus rapide. Le processeur envoie la donnée recherchée à la mémoire associative, et celle-ci répond avec son adresse en quelques cycles d'horloge de la mémoire.

Mémoires associatives /Description

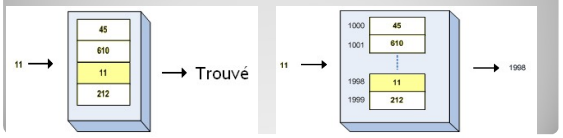
- Une mémoire associative prend en entrée une donnée et renvoie son adresse.
- Le bus de donnée est donc accessible en lecture/écriture, comme sur une mémoire normale.
- Par contre, son bus d'adresse est accessible en lecture et en écriture, c'est une **entrée/sortie**.

1. **L'usage du bus d'adresse comme entrée** est similaire à celui d'une mémoire normale (pour écriture).
2. **L'usage du bus d'adresse comme sortie**, c'est pour le fonctionnement normal de la mémoire associative, càd récupérer l'adresse d'une donnée, demande d'utiliser le bus d'adresse comme une sortie (lecture de l'adresse)



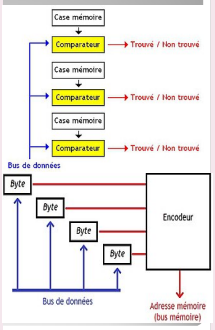
Mémoires associatives /Description

- Lorsque l'on recherche une donnée dans une mémoire CAM, il se peut que la donnée demandée ne soit pas présente en mémoire.
- La mémoire CAM doit donc préciser si elle a trouvé la donnée recherchée avec un signal dédié.
- Un autre problème survient quand la donnée est présente en plusieurs exemplaires en mémoire : la mémoire renvoie alors le premier exemplaire rencontré (celui dont l'adresse est la plus petite).
- D'autres mémoires renvoient aussi les autres exemplaires, qui sont envoyées une par une au processeur.



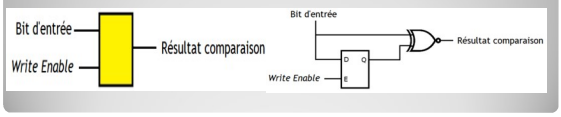
Mémoires associatives /Fonctionnement

- Une mémoire associative est organisée autour de deux fonctions : **vérifier la présence d'une donnée** dans chaque case mémoire (effectuer une comparaison, donc), et **déterminer l'adresse** d'une case mémoire sélectionnée.
- La donnée à rechercher dans la mémoire est envoyée à toutes les cases mémoires simultanément et toutes les comparaisons sont effectuées en parallèle.
- Une fois la case mémoire est identifiée, il faut déduire son adresse. La traduction va se faire par un circuit assez semblable au décodeur : l'**encodeur**.
- Si la donnée est présente en plusieurs exemplaires dans la mémoire, la solution est de sélectionner un seul, généralement celui qui a l'adresse la plus petite (ou la plus grande). Pour cela, l'encodeur doit subir quelques modifications et devenir un **encodeur à priorité**.



Mémoires associatives /Fonctionnement

- Avec une mémoire associative normale, la comparaison réalisée par chaque case mémoire est une comparaison d'égalité stricte : on vérifie que la donnée en entrée correspond exactement à la donnée dans la case mémoire.
- L'interface d'une cellule mémoire associative contient donc deux entrées et une sortie : une sortie pour le résultat de la comparaison, une entrée pour le bit d'entrée, et une entrée *write enable* qui dit s'il faut faire la comparaison avec le bit d'entrée ou écrire le bit d'entrée dans la cellule. Une cellule mémoire de ce genre sera appelée une **cellule CAM** dans ce qui suit.



Mémoires associatives/ Opérations

- **Recherche (Read) :** L'opération de recherche est utilisée pour trouver une donnée. La mémoire renvoie alors l'adresse où la donnée correspondante est stockée.
- **Écriture (Write) :** Stocker une nouvelle donnée dans la mémoire associative.
- **Mise à jour (Update) :** Cette opération consiste à modifier le contenu d'une donnée existante dans la mémoire associative. La recherche est généralement effectuée pour localiser la donnée à mettre à jour.
- **Suppression (Delete) :** L'opération de suppression permet de retirer une donnée de la mémoire associative.
- **Comparaison (Compare) :** Consiste à comparer une partie du contenu avec les données stockées dans la mémoire pour déterminer s'il y a une correspondance. Cela peut être utilisé pour vérifier si une donnée spécifique est déjà présente dans la mémoire.

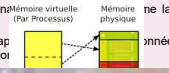
Mémoires associatives/Caractéristiques

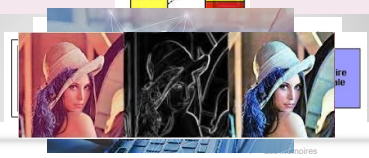
- **Accès basé sur le contenu :** L'accès à l'information se fait en fournissant une partie du contenu de la donnée recherchée plutôt que son emplacement exact.
- **Association de données :** La mémoire associative fonctionne en associant des données entre elles. Lorsqu'une partie du contenu est fournie en tant que requête, la mémoire renvoie l'adresse où cette donnée est stockée.
- **Recherche rapide :** La mémoire associative permet des recherches rapides en se basant sur le contenu. Utile dans des applications où la correspondance de contenu est plus pertinente que l'emplacement physique de la donnée.
- **Applications spécifiques :** Ce type de mémoire est souvent utilisé dans des applications spécifiques, telles que la recherche d'informations basée sur des caractéristiques spécifiques plutôt que sur des adresses mémoire.
- **Complexité et coût :** elle est généralement plus complexe à mettre en œuvre et peut être plus coûteuse que la mémoire conventionnelle.

Mémoires associatives/Applications

Utilisées pour une recherche rapide de l'information

- **Tables de Translation d'Adresses (TLB) :** Dans les systèmes de gestion de mémoire virtuelle, les TLB utilisent souvent des mémoires associatives pour traduire les adresses virtuelles en adresses physiques.
- **Cache Mémoire :** Certains caches mémoire utilisent des mémoires associatives pour accélérer l'accès aux données fréquemment utilisées en évitant de parcourir toutes les adresses.
- **Filtrage de Paquets dans les Routeurs :** Utilisées dans les routeurs pour filtrer les paquets de données en fonction de leur contenu, ce qui peut accélérer le traitement des paquets.
- **Reconnaissance de Motifs :** Reconna… ne la reconnaissance de formes dans des images.
- **Systèmes de Sécurité :** Recherche rap… onnées de signatures de sécurité, contribuant ainsi à la détection d'intrusio…





Mémoires associatives/Applications

- **Base de Données et Recherche d'Information :** Dans les bases de données, les mémoires associatives peuvent accélérer la recherche en permettant des opérations de recherche basées sur le contenu des données plutôt que sur des clés spécifiques.
- **Traitement du Langage Naturel :** Dans certaines applications de traitement du langage naturel, les mémoires associatives peuvent être utilisées pour accélérer la recherche de mots ou de phrases spécifiques.
- **Réseaux de Neurones :** Certains types de mémoires associatives sont utilisés dans les architectures de réseaux de neurones pour stocker et récupérer des motifs spécifiques.
- **Gestion de la File d'Attente :** Dans les systèmes de gestion de file d'attente, les mémoires associatives peuvent être utilisées pour rechercher rapidement des éléments dans la file d'attente en fonction de leur contenu

Mémoires associatives/Exemple

- **Application : Recherche de Filtres de Paquets dans les Routeurs**
- **Description :** Les routeurs dans un réseau informatique sont responsables du transfert des paquets de données d'un réseau à un autre. L'une des tâches critiques des routeurs est de décider comment diriger ces paquets en fonction des adresses IP de destination.
- **Comment cela fonctionne :** La mémoire associative (CAM) est utilisée pour stocker des filtres de paquets, où chaque filtre est associé à une action spécifique à prendre. L'adresse IP de destination d'un paquet entrant est utilisée comme requête pour la mémoire associative. Si une correspondance est trouvée, l'action associée à ce filtre est immédiatement appliquée.
- **Exemple :** Supposons que vous ayez un filtre dans la mémoire associative avec l'adresse IP 192.168.1.1 associée à l'action "Envoyer vers le réseau A". Lorsqu'un paquet arrive avec l'adresse IP de destination 192.168.1.1, la mémoire associative peut rapidement trouver cette correspondance et diriger le paquet vers le réseau A sans avoir besoin de parcourir une table de routage ou de faire des comparaisons séquentielles.
- **Avantages :** Cette utilisation de la mémoire associative permet des recherches très rapides, car la mémoire elle-même est capable de fournir l'adresse de la donnée associée à la requête (dans ce cas, l'action à prendre). Cela accélère considérablement le processus de prise de décision dans les routeurs, ce qui est crucial pour le bon fonctionnement des réseaux, en particulier dans les environnements où la latence doit être minimisée

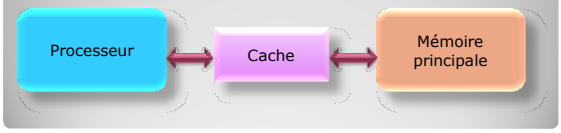
12. Mémoire Cache (Tampon)

Définition

Un cache est une mémoire très rapide de taille restreinte (petite taille) placée entre le processeur et la mémoire principale et dont le temps d'accès est de 4 à 20 fois inférieur à celui de la mémoire centrale.

Rôle ?

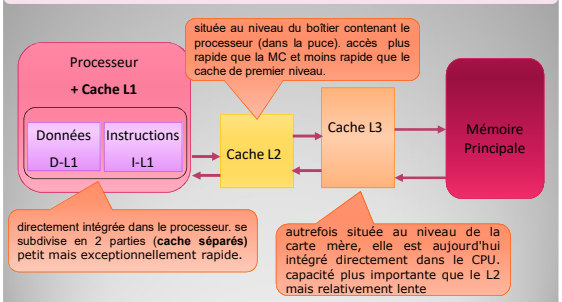
1. Elle contient une copie d'une zone de mémoire centrale
2. permet de diminuer les temps d'accès
3. garder les informations de la mémoire principale dont le processeur se sert le plus souvent à un instant donné.



12.2 Les niveaux de caches

➢ Les mémoires cache peuvent être placés entre tous circuits manipulant des données fonctionnant de façon **asynchrone**, par exemple processeur et mémoire vive, réseau, disque dur, …etc.
➢ La mémoire cache des microprocesseurs est souvent subdivisée en niveaux qui peuvent aller jusqu'à **trois**.
➢ Elle est très rapide, et donc très chère. Il s'agit souvent de SRAM (Static Random Access memory).

12.2 Les niveaux de caches



12.3 Relation entre les niveaux de cache

Les caches sont classées selon leurs contenus en inclusif et exclusif

Cache inclusif

➢ Le contenu du niveau L1 se trouve aussi dans L2 et la taille globale du cache est celle du cache L2.

➢ Le cache inclusif : historiquement le plus utilisé.

Cache exclusif

➢ Le contenu des niveaux L1 et L2 sont différents et la taille globale du cache est la taille L1 + taille L2 (taille plus importante).

➢ Ce cache doit gérer la non duplication des données ce qui prend du temps.

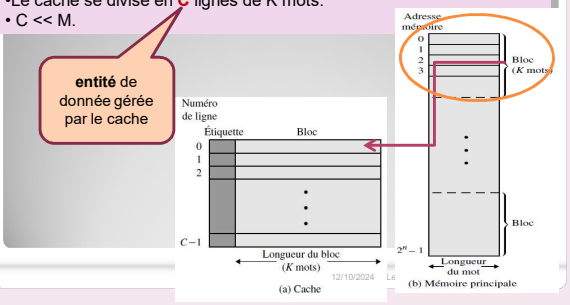
12.4 Structure d'une M cache vs M principale

•Mémoire principale de 2^n mots adressables.

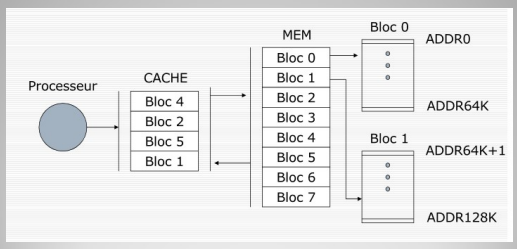
•Correspondance avec la mémoire cache ➡ MC se compose de blocs de K mots donc un nombre de bloc $M = 2^n/K$ blocs.

•Le cache se divise en C lignes de K mots.

• C << M.



12.4 Structure d'une M cache vs M principale



12.4 Structure d'une M cache vs M principale

Une ligne : le plus petit élément de données qui peut être transféré entre la mémoire cache et la mémoire de niveau supérieur.

Un mot : le plus petit élément de données qui peut être transféré entre le processeur et la mémoire cache.

➢ La taille d'un cache se calcul en fonction du nombre de blocs (lignes) du cache et de la taille du bloc (nombre de mots par bloc).

Taille du Cache = Nombre_Bloc x Taille_Bloc

Taille du Bloc = Nombre_Mots (par bloc) x Taille_Mot

Exemple

Taille du bloc = 16 octets (4 mots)

Cache = 16 blocs = 4*4*16 = 256 octets

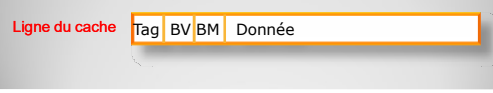
Mémoire = 128 blocs = 4*4*128 = 2KiO

12.5 Calcul des adresses physiques

La présence de la mémoire cache est transparente pour un processeur le CPU demande toujours à accéder à une adresse en mémoire centrale (pour lecture ou écriture)

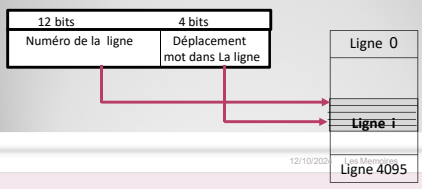
Chaque ligne du cache contient :

- **Une étiquette** (**Tag** en anglais) ou identificateur d'un bloc, composée généralement d'une partie de l'adresse de la mémoire principale, qui identifie le bloc stocké.
- **Bit Valide (BV)** : Si à 1, alors bloc occupé, si 0 bloc vide (libre).
- **Bit Modifié(BM)**: si à 1, alors le bloc a subi une écriture, si non le bloc n'a jamais été modifié depuis sa lecture de la DRAM.
- **La donnée** à stocker: les mots mémoires



Exemple calcul de l'étiquette (Tag)

- On donne la taille du cache est 2048 mots, et la mémoire principale est 64 K mots. La taille de la ligne est 16 mots
- Le **cache** a donc N = 2048/16 = 128 = 2^7 lignes,
- La MP est de 64K/16 = 4K lignes (ou blocs) = 2^{12} blocs donc 12 bits d'adressage des blocs.
- La **taille de la ligne** est 16 mots = 2^4 mots donc 4 bits d'adressage des mots de la ligne.
- L'**étiquette** est donc décomposée de 12 bits (de poids fort) pour l'adressage des blocs en mémoire et 4 bits (de poids faible) pour l'adressage des mots dans le bloc.



12.6 Le mapping

➢ La mémoire cache ne peut pas contenir toute la mémoire principale, il faut définir une méthode indiquant à quelle adresse de la mémoire cache doit être écrite une ligne de la mémoire principale. Cette méthode s'appelle le **mapping**.

Les caches peuvent avoir trois types de mapping:

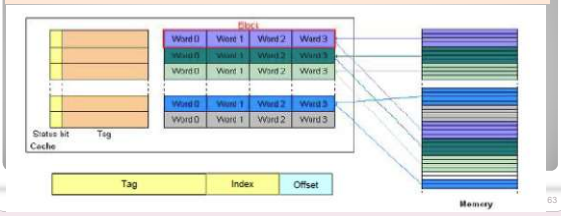
- ➢ **Cache à correspondance directe (*direct mapped*)**
- ➢ **Cache totalement associatif (*fully associative*)**
- ➢ **Cache associatif par ensemble de blocs (*set associative*)**

12.6 .1 Cache à correspondance directe (*direct mapped*)

➢ Chaque ligne de la mémoire principale ne peut être enregistrée qu'à une seule adresse de la mémoire cache.
➢ La sélection de la ligne où la donnée sera enregistrée est obtenue par: Ligne = Adresse mod Nombre de lignes.
➢ Une ligne de cache est partagée par de nombreuses adresses de la mémoire de niveau supérieur. Il nous faut donc un moyen de savoir quelle donnée est actuellement dans le cache.
➢Cette information est donnée par le **Tag**, L'**index** correspond à la ligne où est enregistrée la donnée.
En plus, le contrôleur de la mémoire cache doit savoir si une ligne contient une donnée ou non c'est le role du bit (**BV**)
Offset = n° du mot dans la ligne
A partir de l'adresse d'une ligne en mémoire on sait directement dans quelle ligne du cache elle doit se trouver

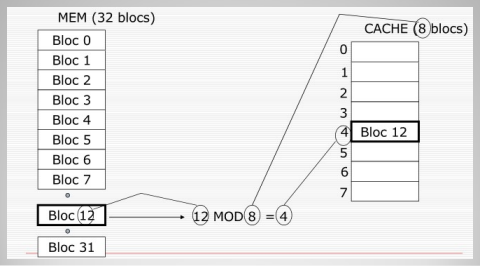
12.6 .1 Cache à correspondance directe (*direct mapped*)

Exemple
Taille Lignes = 32 octets
Capacité Mémoire cache = 512 Ko : 16384 lignes
Capacité Mémoire centrale = 128 Mo : doit être gérée via les 512 Ko de cache et ses 16384 lignes.
Chaque ligne du cache correspond à : 128 x 1024 x 1024 / 16384 = 8192 octets = 256 lignes
Une ligne ***i*** du cache contiendra à un instant donné une des 256 lignes ***j*** de la mémoire tel que ***i* = *j* mod 16384**



12.6 .1 Cache à correspondance directe (*direct mapped*)

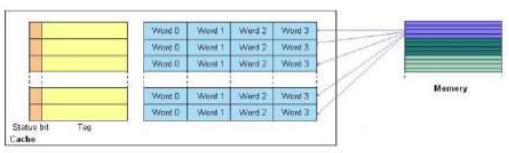
Exemple



12.6.2 Cache totalement associatif (*fully associative*)

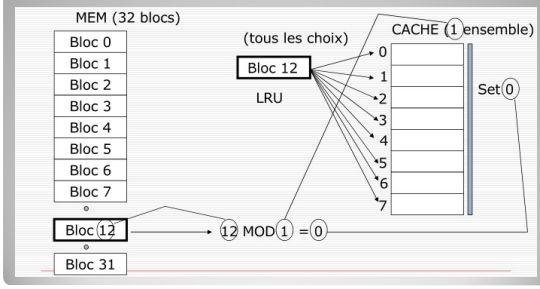
Un bloc de la mémoire de niveau supérieur peut être placé n'importe où dans le cache. Cette méthode requiert beaucoup de logique car elle donne accès à de nombreuses possibilités. Ceci explique que l'associativité complète n'est utilisée que dans les mémoires cache de petite taille.

Cela donne le format suivant de l'adresse :



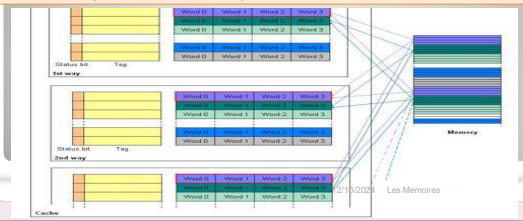
12.6.2 Cache totalement associatif (*fully associative*)

Exemple



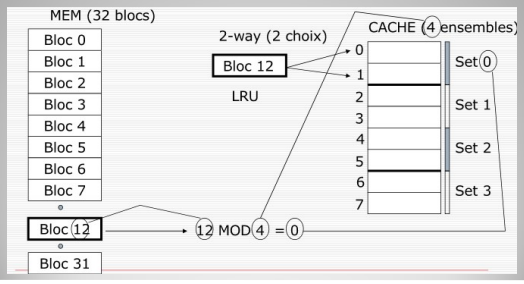
12.6.3 Cache associatif par ensemble de blocs (*set associative*)

➢ Un bloc peut être placé dans un ensemble restreint de places. C'est un compromis entre le mapping direct et complètement associatif combinant la simplicité de l'un et l'efficacité de l'autre.
➢ La mémoire cache est divisée en ***ensembles* (*sets*)** de N lignes de cache. Une ligne de la mémoire de niveau supérieur est affectée à un ensemble, elle peut par conséquent être écrite dans n'importe laquelle des voies. Ceci permet d'éviter de nombreux défauts de cache conflictuels.
➢ En général, la sélection de l'ensemble est effectuée par: *Ensemble = Adresse mémoire* mod (*Nombre d'ensembles*).



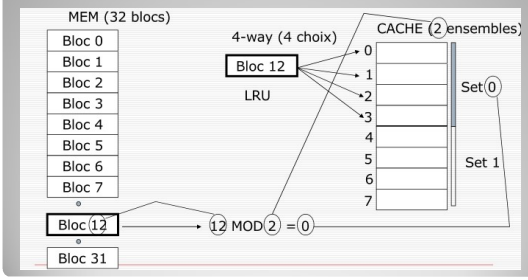
12.6.3 Cache associatif par ensemble de blocs (*set associative*)

Exemple 1 : Cache 2-way set associative



12.6.3 Cache associatif par ensemble de blocs (*set associative*)

Exemple 2: Cache 4-way set associative



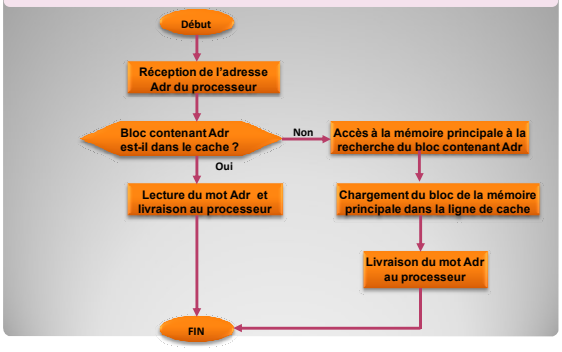
12.7 Remplacement d'une ligne de cache

➢Les caches associatifs de N voies et complètement associatifs impliquent le mapping de différentes lignes de la mémoire de niveau supérieur sur le même ensemble.
➢Ainsi, il faut désigner la ligne qui sera effacée au profit de la ligne nouvellement écrite. Le but de l'algorithme de remplacement des lignes de cache est de choisir cette ligne de manière optimale.
➢ Il existe 3 méthodes pour choisir la ligne à remplacer
Remplacement Aléatoire : Simple à mettre en œuvre mais peu efficace car peut supprimer des lignes très accédées
Remplacement de la plus ancienne (LRU : Last Recently Used) : qui mémorise la liste des derniers éléments accédés. Nécessite des compteurs associés aux lignes
Remplacement de la moins utilisée (LFU : Least Frequently Used) **:** Nécessite également des compteurs

12.8 Fonctionnement des Caches

➢ Lors d'une requête à une donnée (lecture), la présence de cette donnée dans le cache est testée. Si la donnée est présente (**succès de cache**), elle est lue à partir du cache. Si elle est absente (**défaut de cache**), la zone de la mémoire centrale à laquelle elle appartient est chargée dans le cache (dans une zone libre ou en remplacement d'une zone non utilisée).
➢ A chaque lecture dans la mémoire centrale, le processeur recopie dans la mémoire cache le contenu de la zone voisine des données dont il a besoin.
➢ A la lecture suivante, il y a de très fortes probabilités que les données dont il a besoin se trouve dans la mémoire cache. Ce mécanisme permet d'accélérer la vitesse du processeur.
➢ Le processeur génère l'adresse, Adr, d'un mot à lire. Si celui-ci se trouve dans le cache, il est fourni au processeur. Dans le cas contraire, on charge le bloc contenant ce mot dans le cache et on le transmet au processeur.

12.8 Fonctionnement des Caches



12.9 Les défauts des Caches

Il existe trois types de défauts de cache en système monoprocesseur :
➢ Les **défauts de cache obligatoires** (compulsory): ils correspondent à la première demande du processeur pour une donnée/instruction spécifique et ne peuvent être évités car au premier accès le bloc n'est pas dans le cache, et il doit être transféré dans le cache; ils sont aussi appelés échecs de démarrage à froid.
➢ Les **défauts de cache capacitifs** (*capacity*): Si le cache ne peut contenir tous les blocs nécessaires pendant l'exécution d'un programme, il ne peut donc pas contenir toutes les données nécessaires, des échecs de capacité interviendront à cause de blocs écartés.
➢ Les **défauts de cache conflictuels** (*conflict)*: Si la stratégie de placement des blocs est associative par ensemble ou à correspondance directe, les échecs de conflit interviennent parce qu'un bloc peut être écarté et rappelé plus tard si trop de blocs doivent être placés dans son ensemble.

12.9 Localité et pre-fetching

➢ Pour améliorer le taux de succès du cache le processeur effectuera un chargement en avance des données dont il aura besoin. Cette opération est nommée ***Pre-fetching***.
➢ Les Algorithmes de pre-fetching sont basés sur les principes de **localité**.
➢ Il existe deux principe de localités sur lesquels se base le processeur pour la recherche des blocs à mettre dans le cache.

1.Le principe de localité spatiale : qui indique que l'accès à une instruction située à une adresse X va probablement être suivi d'un accès à une zone tout proche de X.

2.Le principe de localité temporelle : qui indique que l'accès à une zone mémoire à un instant donné a de fortes chances de se reproduire dans la suite du programme.

12.10 Ecriture dans le cache

➢ Quand une donnée/instruction se situe dans le cache, les systèmes en possède deux copies: une dans la mémoire de niveau supérieur et une dans la mémoire cache.
➢ Deux différentes politiques s'affrontent:
Ecriture simultanée (write through): la donnée/instruction est écrite à la fois dans le cache et dans la mémoire de niveau supérieur.
Écriture différée ou réécriture (write back): l'information est écrite uniquement dans la ligne du cache. Quand cette ligne est supprimée du cache, on l'écrit en mémoire centrale (de niveau supérieur). Cette technique est la plus répandue car elle permet d'éviter de nombreuses écritures mémoires inutiles.
➢ Cependant, afin de ne pas écrire des informations qui n'ont pas été modifiées, chaque ligne de la mémoire cache, est pourvue d'un bit indiquant si elle a été modifiée.

12.12 Taille des caches

➢ Malgré la taille importante de la mémoire principale (de quelques centaines de Mio à quelques Gio) mais son débit reste très lent par rapport au débit requis par le processeur.
➢ On accélère la vitesse de lecture des informations par le CPU en les plaçant (en avance) dans le cache caractérisé principalement par :
✓Mémoire associative
✓De type SRAM car doit être rapide
✓Taille : de quelques centaines de Kio à quelques Mio
➢Au fil de leur évolution, les systèmes se sont dotés d'une mémoire cache de taille toujours plus importante, toujours plus rapide et toujours placée le plus proche possible des unités de traitement du processeur.

12.12 Taille des caches

✓ Pour une famille de processeurs, la taille des caches L1 et L2 ne varie généralement pas.
✓ La taille du cache L3 est souvent proportionnelle au nombre de cores. Plus ils sont nombreux, plus ce cache est de taille.
✓ Dans les caractéristiques des processeurs, c'est le **cache de troisième niveau** (L3) qui est mentionné.

Exemple
Tous les processeurs Core i3 ont par exemple **3 Mio de cache**.

Exemple 1 : comparaison des tailles des caches de quelques processeurs
- Tous les processeurs Core i3 ont par exemple **3 Mio de cache**.
- Les Core i5, **6 Mio de cache**.
- Les Core i7, **8 Mio de cache**

Exemple 2 : comparaison vitesse Mémoire centrale Vs Cache
Processeur : AMD Athlon XP 2200+ (1.8 Ghz)
Mémoire : SDRAM-DDR 2100
Mémoire Centrale
Fréquence de 133 Mhz et mots de 64 bits
Débit théorique maximum de 2,1 Go/s
Cache Processeur
Cache L1 du processeur : débit mesuré de 18 Go/s
Cache L2 du processeur : débit mesuré de 5,6 Go/s

Remarque

Remarque
Les performances globales de la mémoire cache dépendent de plusieurs facteurs corrélés

1. Taille de la mémoire cache et organisation des niveaux
2. Méthode d'association des lignes entre mémoire centrale et cache
3. Rapidité d'accès à la bonne ligne dans le cache
4. Méthode de remplacement des lignes
5. Algorithmes de *pre-fetching*

Dans le but d'avoir un taux de succès global qui doit être le plus élevé possible tout en assurant un temps d'accès bas autour de 90%.

Processeur	Type	Année de mise en service	Cache L1	Cache L2	Cache L3
IBM 360/85	Gros ordinateur	1968	16 à 32 Ko	—	—
PDP-11/70	Mini-ordinateur	1975	1 Ko	—	—
VAX 11/780	Mini-ordinateur	1978	16 Ko	—	—
IBM 3033	Gros ordinateur	1978	64 Ko	—	—
IBM 3090	Gros ordinateur	1985	128 à 256 Ko	—	—
Intel 80486	PC	1989	8 Ko	—	—
Pentium	PC	1993	8 Ko/8 Ko	256 à 512 Ko	—
PowerPC 601	PC	1993	32 Ko	—	—
PowerPC 620	PC	1996	32 Ko/32 Ko	—	—
PowerPC G4	PC/serveur	1999	32 Ko/32 Ko	256 Ko à 1 Mo	2 Mo
IBM S/390 G4	Gros ordinateur	1997	32 Ko	256 Ko	2 Mo
IBM S/390 G6	Gros ordinateur	1999	256 Ko	8 Mo	—
Pentium 4	PC/serveur	2000	8 Ko/8 Ko	256 Ko	—
IBM SP	Serveur haut de gamme/ superordinateur	2000	64 Ko/32 Ko	8 Mo	—
CRAY MTA	PC/serveur	2001	16 Ko/16 Ko	96 Ko	4 Mo
Itanium	PC/serveur	2001	16 Ko/16 Ko	96 Ko	4 Mo
SGI Origin 2001	Serveur haut de gamme	2001	32 Ko/32 Ko	4 Mo	—