ARCHITECTURE DES ORDINATEURS

PARTIE 2

5 DMA

6 PIPELINE

7 MEMOIRE CACHES

8 DS

GESTION DES ENTRÉES SORTIES

TROIS METHODES DE GESTION DES ENTRÉES -SORTIES

- ♣ L'attente active ou la scrutation (spolling)
- ♣ Les entrées-sorties pilotées par les interruptions
- ♣ L'utilisation d'un dispositif permettant des accès directs à la mémoire, DMA

LA SCRUTATION OU SPOLLING

Vérification répétitive de l'état d'un ou plusieurs éléments d'un système pour y détecter un changement.

INTERRUPTION

Rôle d'une interruption: Consiste à interrompre temporairement le programme actuel pour exécuter un autre qui est urgent.

Il existe deux types:

Interruption Masquable : On peut l'ignorer .

Exemple : Brancher une USB .

Interruption Non Masquable : C'est urgent , on ne peut pas l'ignorer .

Exemple : Sortir d'une boucle infini.

Quelques Notions:

Pour gérer les interruptions le contrôleur d'interruption se charge de ça .

LE PIC (Contrôleur Programmable d’Interruptions)

- Établir des priorités entre les différentes sources .
- Décharger le CPU de la détermination de la source d'une interruption, quand celle-ci se présente .

LE VECTEUR DES INTERRUPTIONS

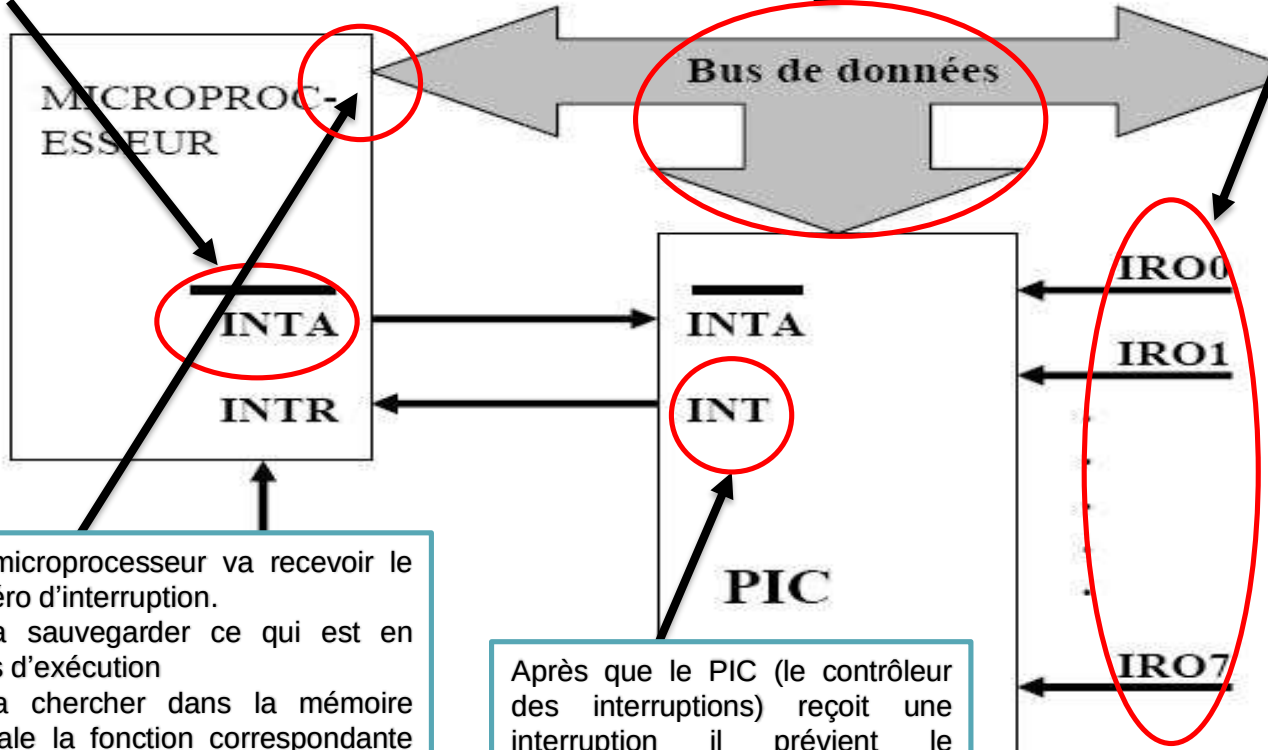
C’est une table dans la mémoire centrale qui fait la correspondance entre les numéros des interruptions et les sous-programmes qui leur convient

GÉRER LES INTERRUPTIONS :

Le MicroProc accepte s'il envoie un 0 dans cette borne « INTA » sinon refuse s'il envoie 1

Si le PIC reçoit une acceptation il lui envoie le code d'interruption qu'il a reçu sur ses bornes IRQ

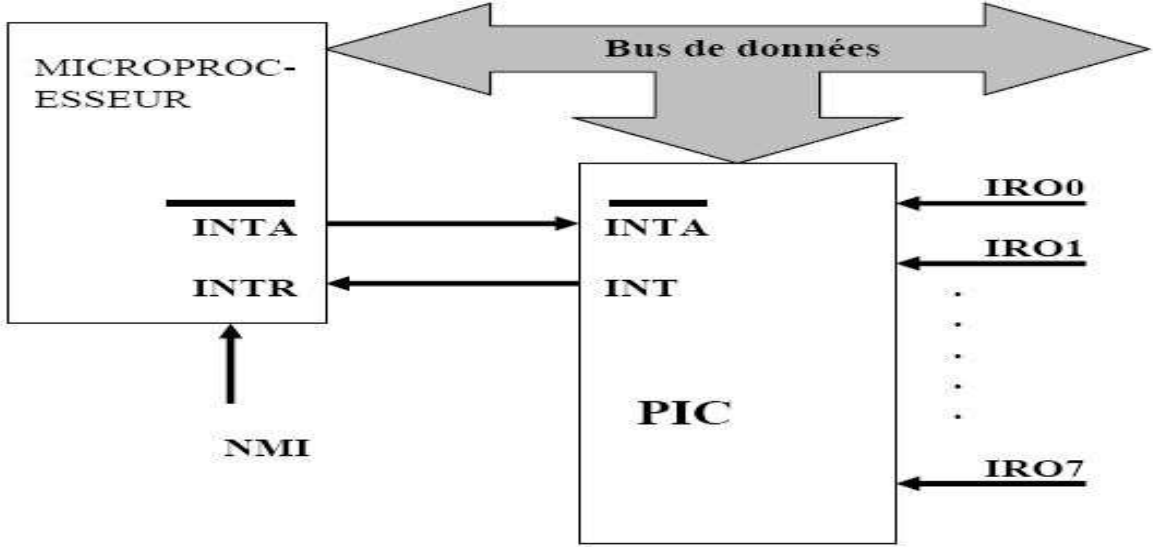
Une interruption arrive sur une de ces lignes qu'on appelle IRQ, la première IRQ0 possède la plus grande priorité et ainsi de suite



-Le microprocesseur va recevoir le numéro d'interruption.
-Il va sauvegarder ce qui est en cours d'exécution
-Il va chercher dans la mémoire centrale la fonction correspondante au numéro qu'il a reçu puis il va l'exécuter.

Après que le PIC (le contrôleur des interruptions) reçoit une interruption il prévient le Microprocesseur par la borne « INT »

GÉRER LES INTERRUPTIONS :



– INTR (*Interrupt Request*) : Utilisée pour indiquer au processeur l'arrivée d'une interruption masquable.

– NMI (*No Masquable Interrupt*) : Utilisée pour envoyer au processeur une interruption non masquable,

– INTA (Interrupt acknowledge) : indique que le microprocesseur accepte l'interruption.

DMA

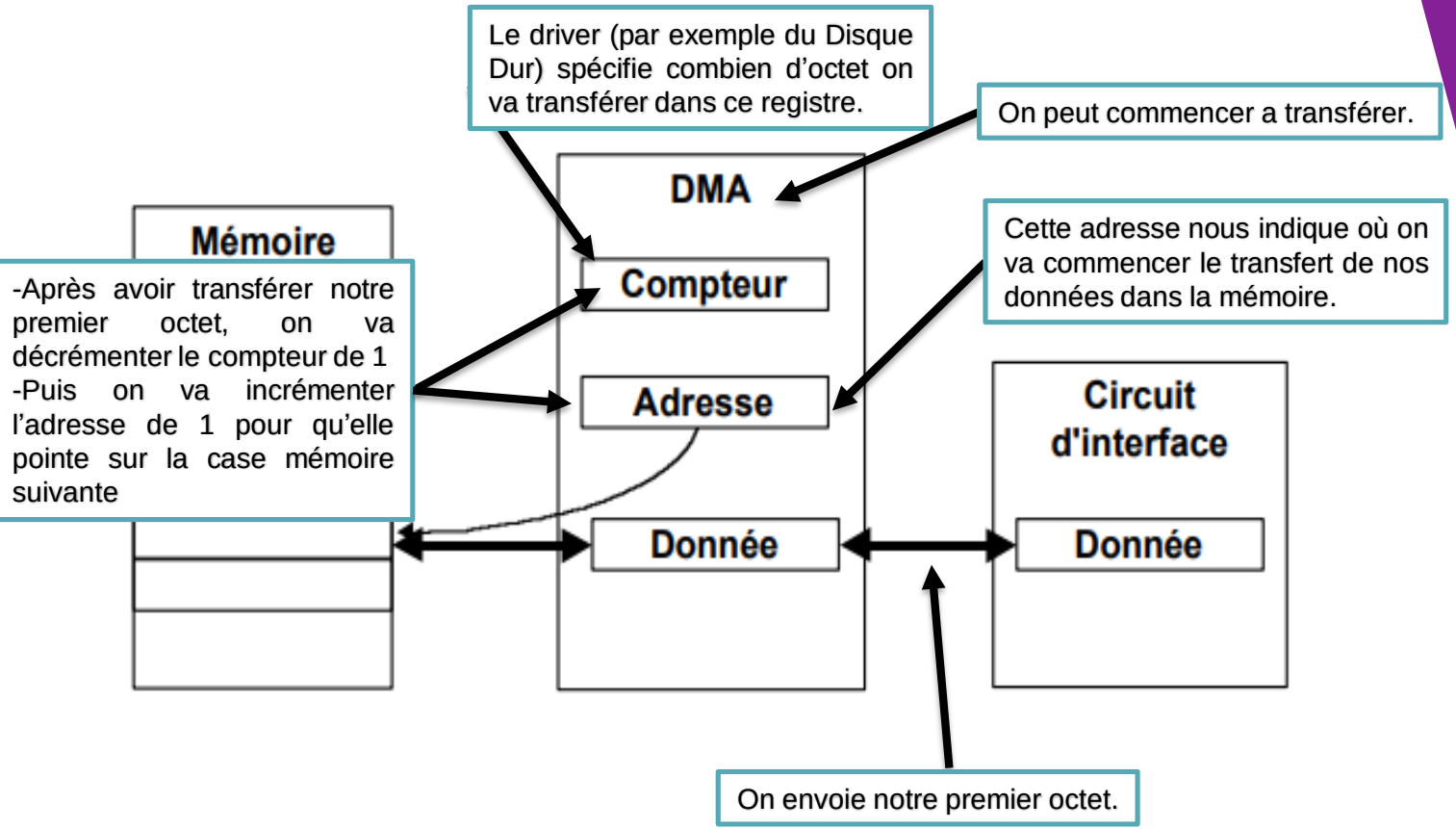
DMA : DIRECT MEMORY ADRESS

C’est quoi : Il s’agit d’un transfert direct entre un périphérique et la mémoire centrale via un contrôleur.

Son Rôle : C’est un transfert sans intervention du microprocesseur, c’est-à-dire le microprocesseur peut faire d’autre travail plus intéressants que de transférer des données.

DMAC : C’est le périphérique qui se charge d’effectuer ce transfert.

LE CONTRÔLEUR DMA



Comment le DMA gère les transferts

- Le driver du périphérique est informé que le transfert de données à la mémoire tampon commencera à l'adresse X.
- Le driver de périphérique ordonne au contrôleur de ce périphérique de transférer la quantité C octet du périphérique à cette adresse.
- Le contrôleur de périphérique initialise ce transfert .
- Le transfert part dans le registre donnée de DAMC.
- DMAC transfert les octets de son registre données dans la mémoire tampon, en incrémentant l'adresse à chaque transfert,
- Le registre compteur contient la quantité d'octet à transférer . Il décrémente à chaque octet transféré.

Mode d’opération de la DMA

Le bus de données peut être utilisé que par un seul périphérique c’est pourquoi on utilise l’un de ces méthodes de partage de bus.

Rafale (Burst) : Le contrôleur DMA (DMAC) conserve le bus de données jusqu’à la fin du transfert.

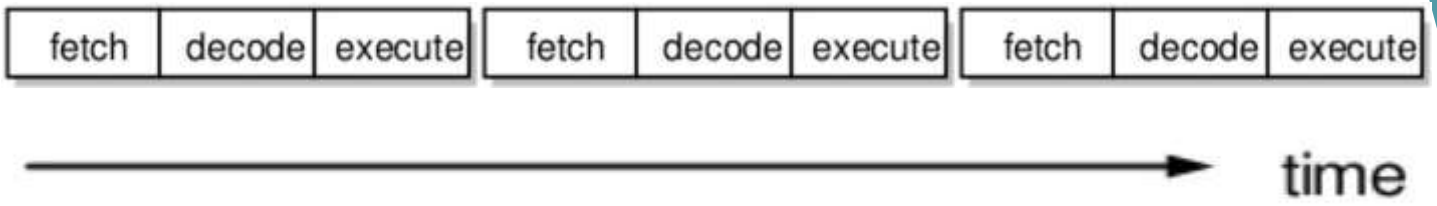
Vol de cycle (Cycle Stealing) : Le DMAC et le processeur se partagent le bus alternativement.

Transparent : Le processeur conserve le bus de données jusqu’à ce qu’il finit.

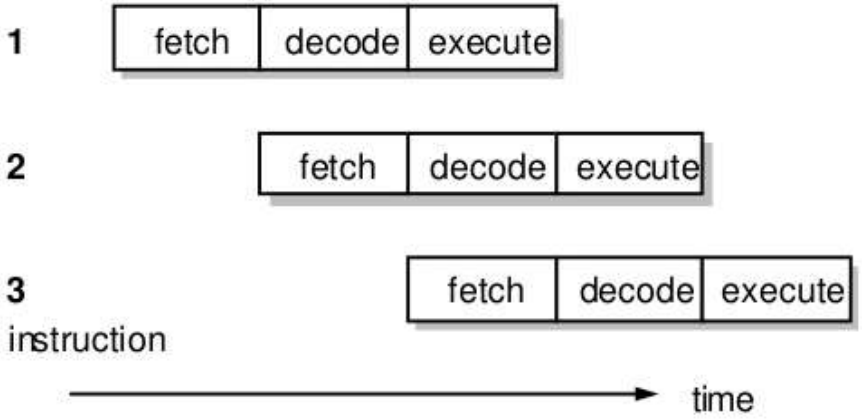
PIPELINE

PIPELINE

SANS Pipeline



AVEC Pipeline



PIPELINE

C’est quoi: c’est un technique qui permet au processeur d’exécuter une nouvelle instruction sans attendre que la précédente soit terminée.

Avantage du pipeline:

- Meilleure utilisation des différentes unités
- Gagner en performances, en exécutant les instructions parallèlement.

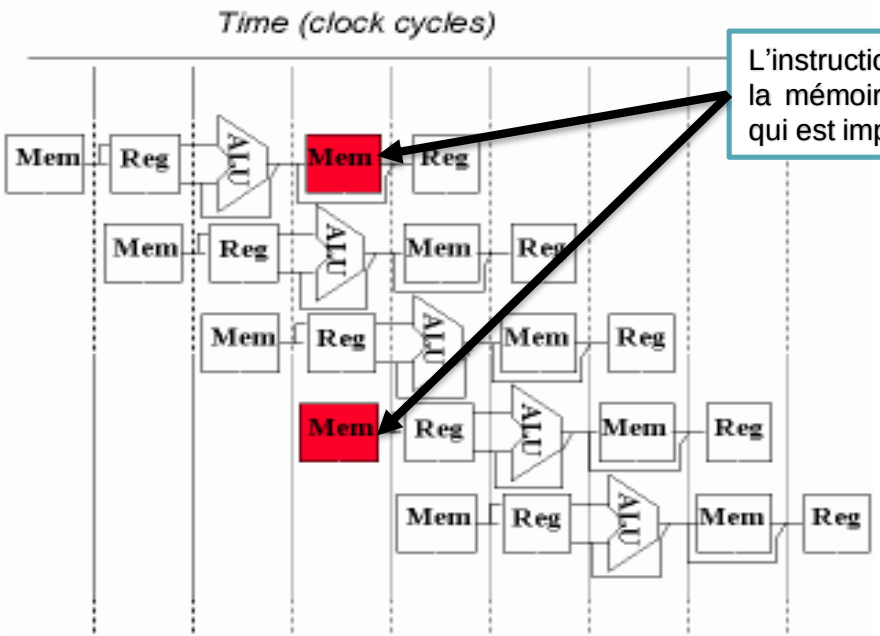
Inconvénient du pipeline

- Plus d'énergie
- Les problèmes de dépendances

PIPELINE PROBLEME 1

Dépendances structurelles

Il arrive que dans certains cas bien précis, plusieurs étages du pipeline aient besoin d'accéder à la même ressource matérielle. Cette ressource peut être la mémoire, un registre, une unité de calcul, ou tout autre chose encore.



Solutions:

- **Attendre que l'unité soit disponible** en retardant l'exécution : peu efficace.
- **Dupliquer la ressource matérielle**

PIPELINE PROBLEME 2

Dépendances de données

Il arrive que dans certains cas bien précis, une instruction essaie de lire une données qui n'est pas encore prête. Prenons l'exemple suivant:

Exemple:
Les deux instructions suivantes vont être exécuter en parallèle selon le principe du pipeline. **Le problème** c'est que l'instruction 2 va faire une opération sur le registre 1 qui n'est pas encore opérer par l'instruction 1.

```
ADD R1,R2,R3        //Cette instruction va mettre la somme de R3 avec R2 dans R1

SUB R4,R1,R3        //Cette instruction va mettre la soustraction du registre R1 avec R3 dans R4
```

PIPELINE PROBLEME 2

Dépendances de données : Solutions

- Arrêter le calcul de R4 tant que R1 n'est pas connu
- Changer l'ordre d'exécution des instructions : réordonnancement (réalisé soit à la compilation, soit par le processeur à la volée)

PIPELINE PROBLEME 3

Dépendances de contrôle

Il arrive dans les cas où il y a une condition à vérifier. Dans ce cas les instructions après la vérification de la condition ne doivent pas être exécuté que si la condition est vérifié. Prenons l'exemple suivant/

Exemple :

```
if (a==3){                    Instruction 1
        instr2;   Instruction 2
        inst3;    Instruction 3
}
```

Le pipeline va exécuter les instructions 1, 2, 3 sans vérifier la condition de if, ce qu'il ne faut pas faire.

PIPELINE PROBLEME 3

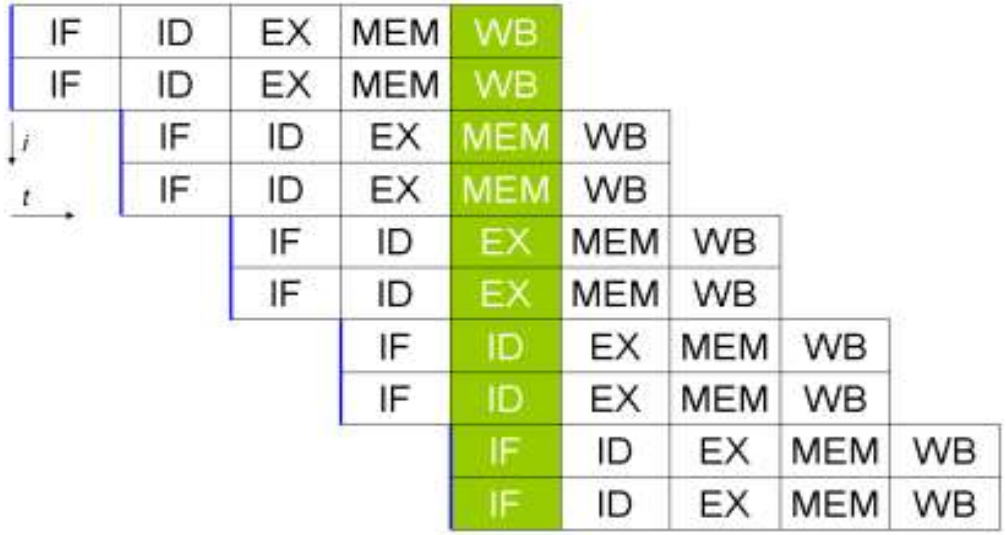
Dépendances de contrôle : Solutions

- Dupliquer l'architecture pour traiter les deux cas du branchement (condition vérifié ou non)
- Attendre l'instruction qu'elle conclue.

Améliorer le pipeline

Architecture superscalaire

Cette astuce consiste à augmenter le nombre d'unités de traitement afin de traiter plusieurs instructions par cycle.



Améliorer le pipeline

VLIW: Very Large Instruction Word

Le séquencement et l'ordonnancement des opérations sont réalisés par le compilateur.

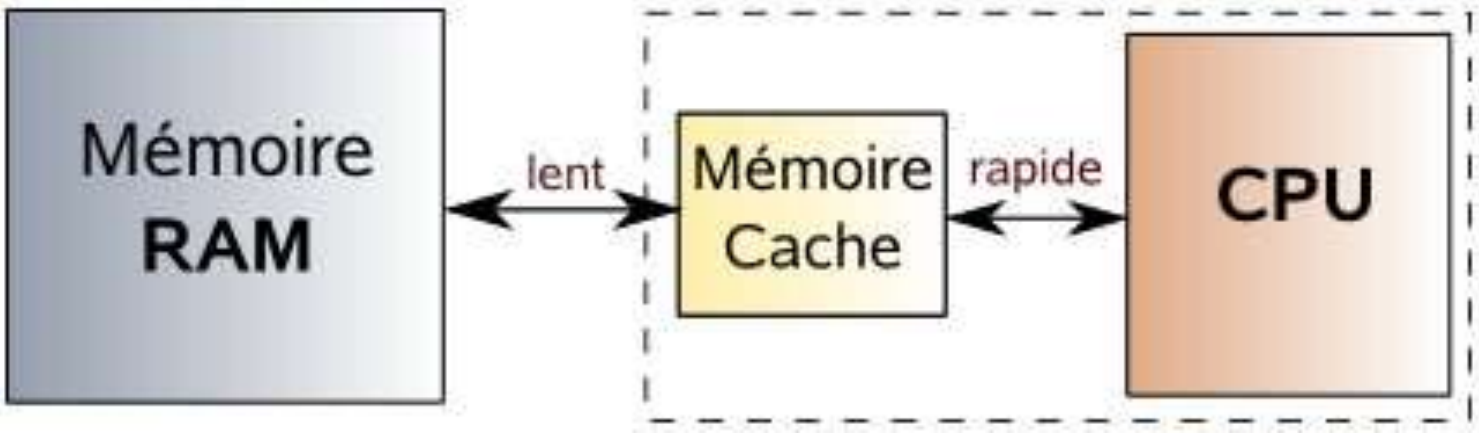
- Le compilateur est responsable de présenter au matériel des instructions exécutables en parallèle
- Contrôle des dépendances de données et de contrôle par le compilateur
- Solution pratique aux problèmes de dépendances dans les superscalaires.

MEMOIRES CACHES

MEMOIRES CACHES

A Quoi Il Sert ????

Sert à stocker des instructions et des données provenant de la mémoire centrale, afin de diminuer le temps d'un accès ultérieur.



MEMOIRES CACHES

Principe de localité

Les programmes possèdent deux caractéristiques intéressantes :

- **localité spatiale :** Ils tendent à utiliser les instructions et les données qui sont situées dans la zone mémoire proche des données et instructions accédées récemment.

- **localité temporelle :** Ils tendent à réutiliser les données et instructions utilisées dans le passé.

```
int SumVec (int *Vec, int N) {

int i, sum = 0;

for (i = 0; i < N; i++)

sum += Vec[i];

return sum;
```

Cette instruction, va être utiliser fréquemment donc on va la mettre en mémoire cache, pour éviter à chaque fois d'aller à la mémoire centrale. **C'est la localité temporelle.**

La fonction for à tendance d'utiliser l'instruction suivante, donc on va les mettre tous les deux dans la mémoire caches. **C'est la localité spatiale**

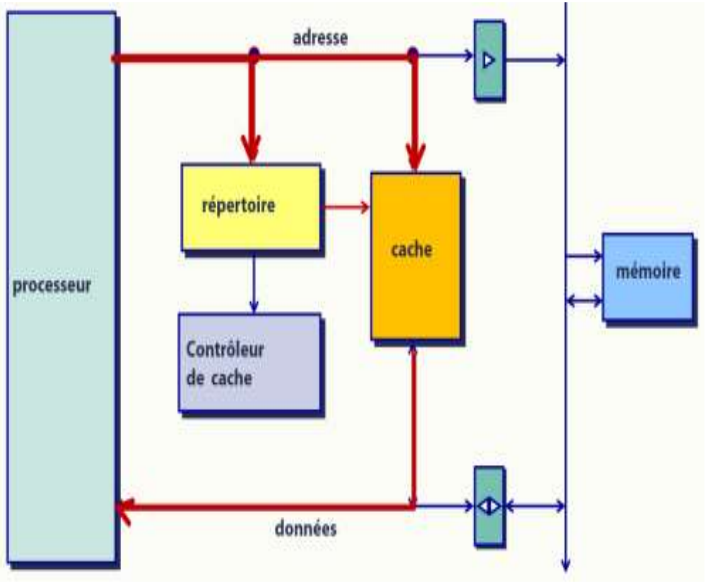
MEMOIRES CACHES

Hiérarchie de mémoire

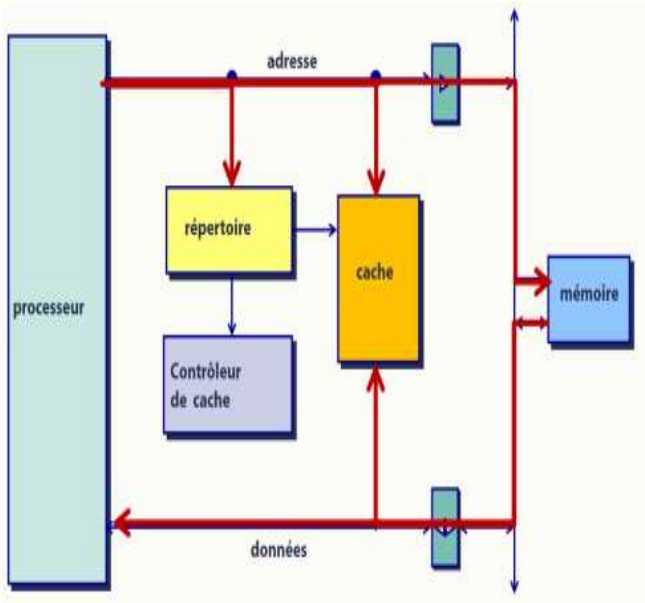
- **Hiérarchie de Mémoire :**
- Les systèmes informatiques ont généralement une hiérarchie de mémoire avec plusieurs niveaux de caches (L1, L2, et parfois L3), en plus de la mémoire principale (RAM) et du stockage de masse.
- Les caches de niveau inférieur (L1) sont plus petits et plus rapides, tandis que les caches de niveau supérieur (L2, L3) sont généralement plus grands mais plus lents en comparaison.
- Les processeurs accèdent d'abord à la mémoire cache L1, puis à L2 et éventuellement à L3 avant d'atteindre la mémoire principale. Cette hiérarchie permet d'optimiser la rapidité d'accès en stockant les données les plus fréquemment utilisées dans les niveaux de cache les plus rapides.

MEMOIRES CACHES

HIT LECTURE

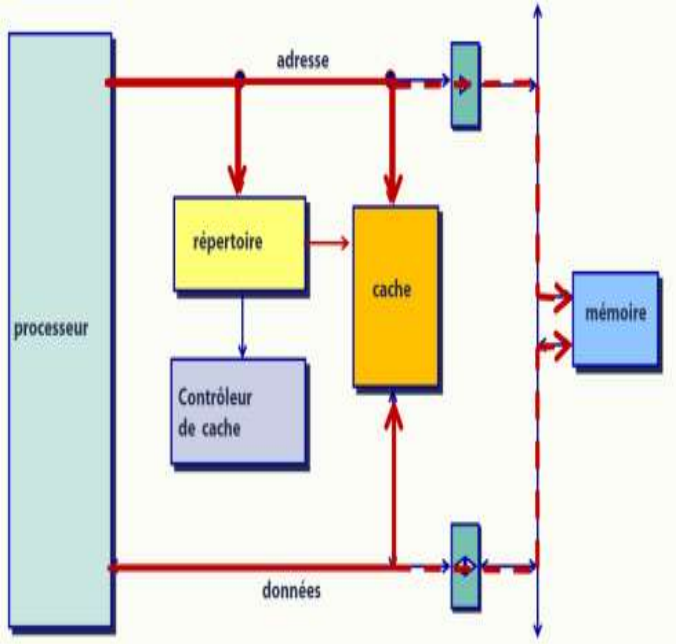


MISS LECTURE

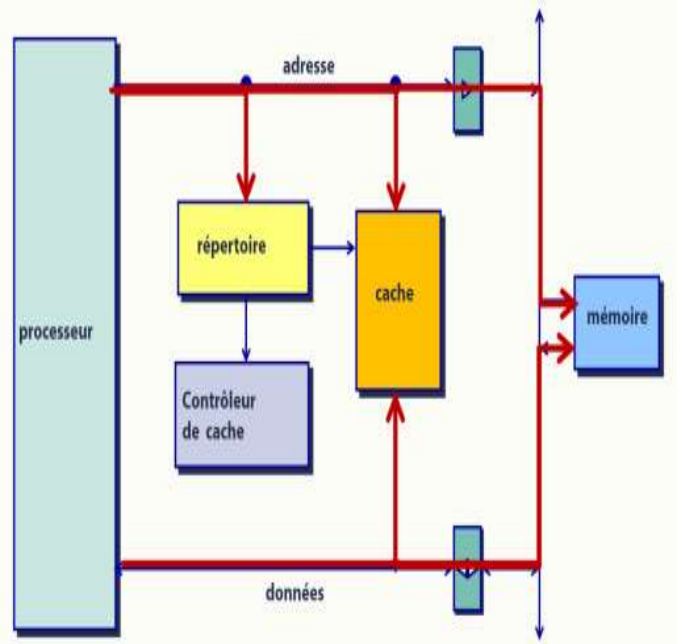


MEMOIRES CACHES

HIT ECRITURE



MISS ECRITURE



Lorsque le mot est présent dans le cache il est modifié dans le bloc du cache et modifié aussi dans la MÉMOIRE CENTRALE

MEMOIRES CACHES

Fonction de correspondance

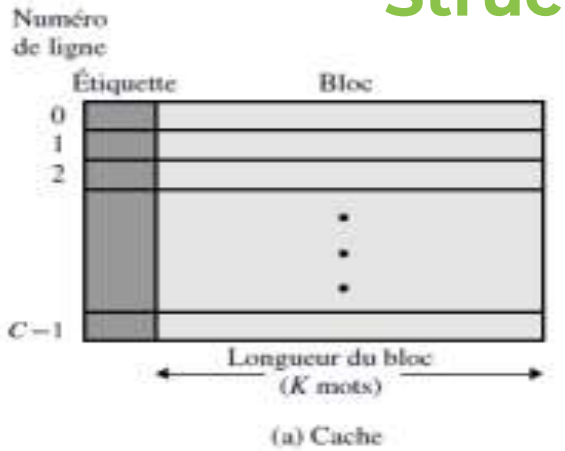
Pour écrire ou lire une donnée depuis la mémoire cache il nous faut une technique, il y a trois différents types de mémoires caches. Donc chacune utilise une technique pour accéder à la donnée. Ces techniques dont ce qu'on appellent fonction de correspondance.

Il existe trois types :

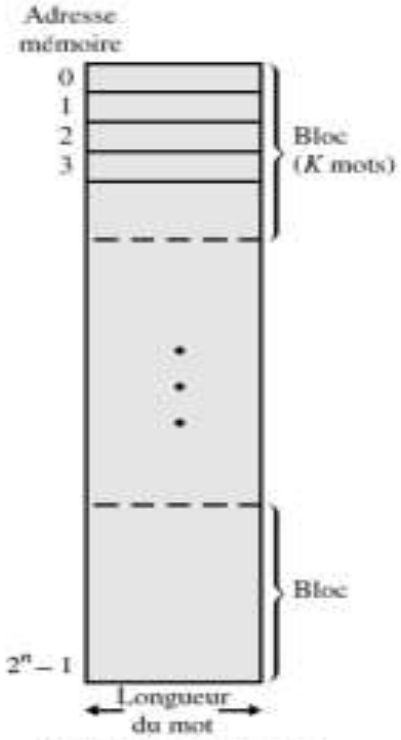
- Les mémoires caches directes .
- Les mémoires caches complètement associatives .
- Les mémoires caches N-associatives .

MEMOIRES CACHES

Structure

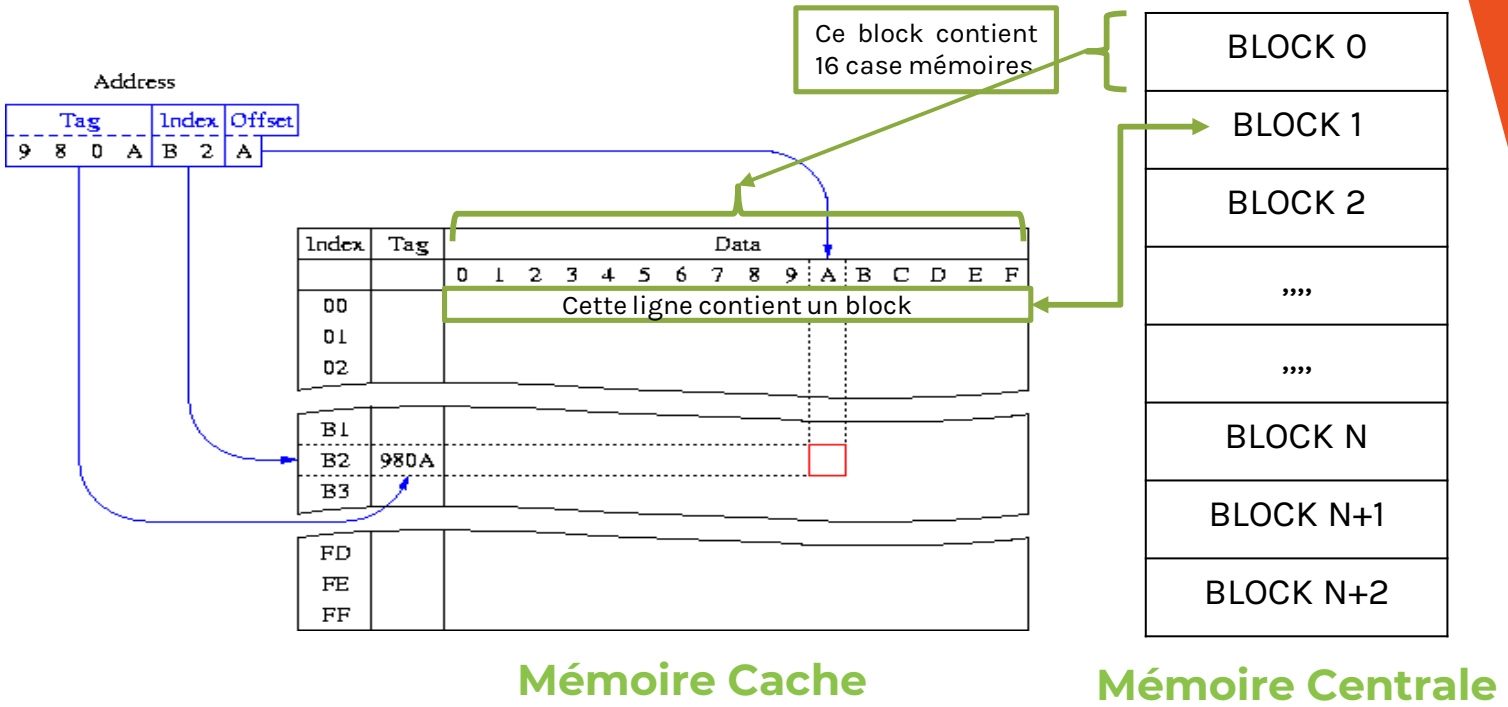


(a) Cache



(b) Mémoire principale

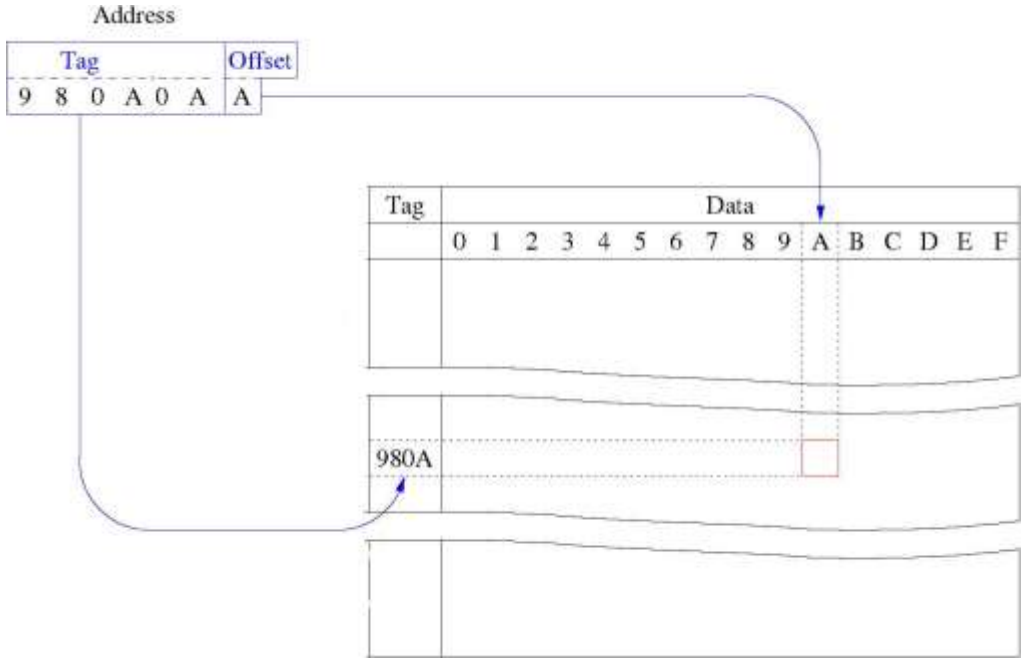
CACHES DIRECTES



Mémoire Cache **Mémoire Centrale**

CACHES ASSOCIATIF

Ce type utilise seulement des étiquettes



CACHES K-ASSOCIATIF

Ce type est identique au cache directe, mais peut contenir plus de block dans une seule ligne