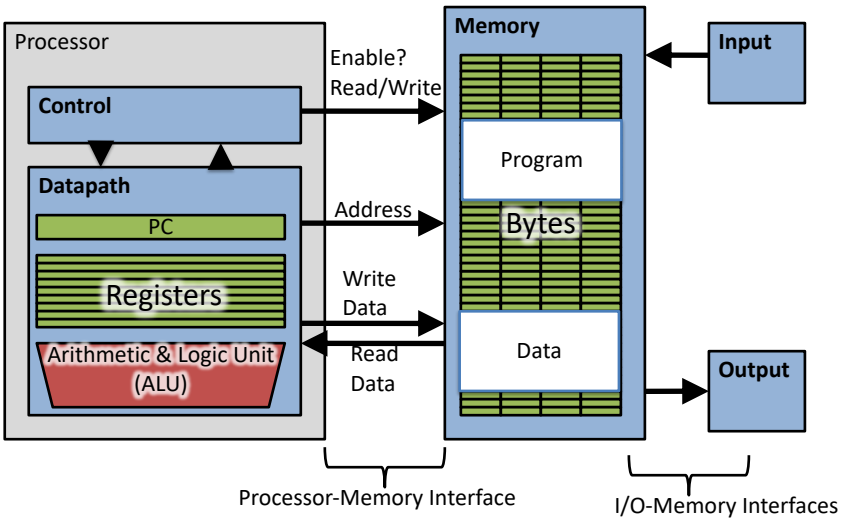
Architecture des Processeurs

Gestion de la mémoire

Components of a Computer

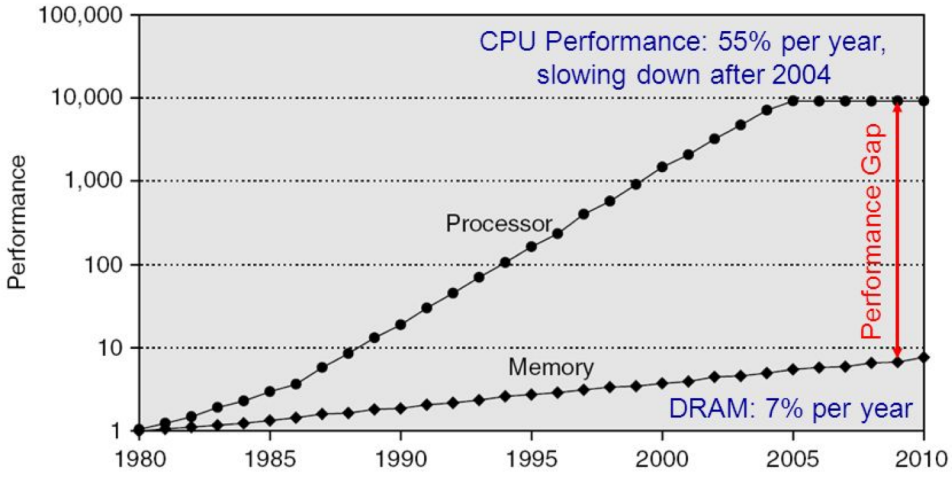


3

Performance

- Le temps passé à attendre une réponse de la mémoire (suspension mémoire ou attente mémoire) a un impact fondamental sur le temps d'exécution d'un programme:
  - Temps d'exécution = (# cycles d'exécution + # cycles d'attente mémoire) × Temps de cycle
  - La pénalité d'accès est le temps (nombre des cycles) nécessaire pour transférer une donnée de la mémoire au processeur.
  - Cycles d'attente mémoire = # accès x pénalité d'accès
  - Pénalité d'accès = # instructions × # accès par instruction × Pénalité d'accès
- Le cache est un moyen de réduire la pénalité d'accès.

Processeur vs DRAM (Latence)



En 1980 un microprocesseur exécute une instruction pendant un accès en DRAM

La lenteur des accès DRAM a un impact très fort sur les performances du système!

En 2017 un microprocesseur exécute 1000 instructions pendant un accès en DRAM

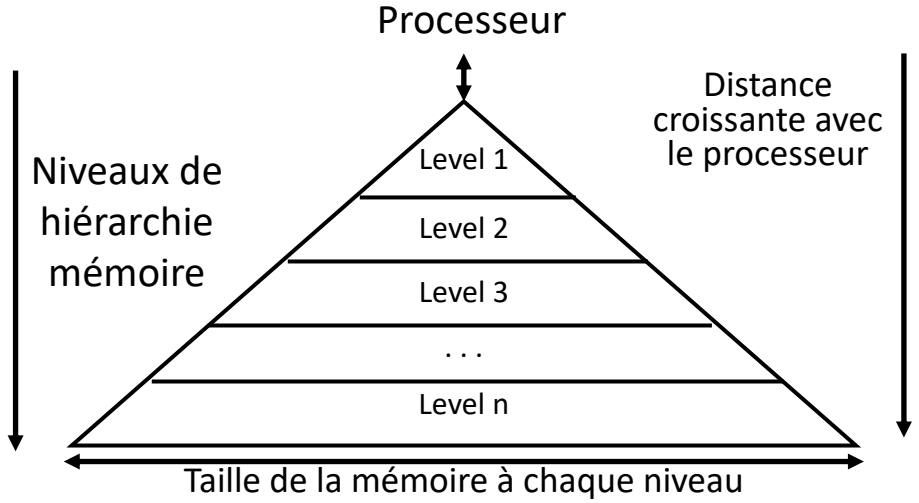
4

Que faire pour lutter contre les problems de latence?

- Vous devez faire une étude bibliographique sur un sujet
  - Vos recherches à la B.U. vous indiquent une dizaine de livres intéressant sur le sujet
  - Pb la BU est à l'autre bout de la ville et l'enregistrement d'un livre à la BU est très long…
  - Qu'est ce que vous faites?
    - Vous allez une seule fois à la BU, prenez le max de livres autorisé, les deposez sur votre bureau et vous si vous avez besoin d'y accèder l'accès sera rapide.
    - C'est exactement le fonctionnement des mémoires caches…
    - Quelque problème (comment optimizer le max? comment choisir les livres dont la probabilité d'être utiles est maximale?

5

Hiérarchie Mémoire



Plus on s'éloigne du processeur plus le bit de mémoire est chère et plus son temps d'accès est long

6

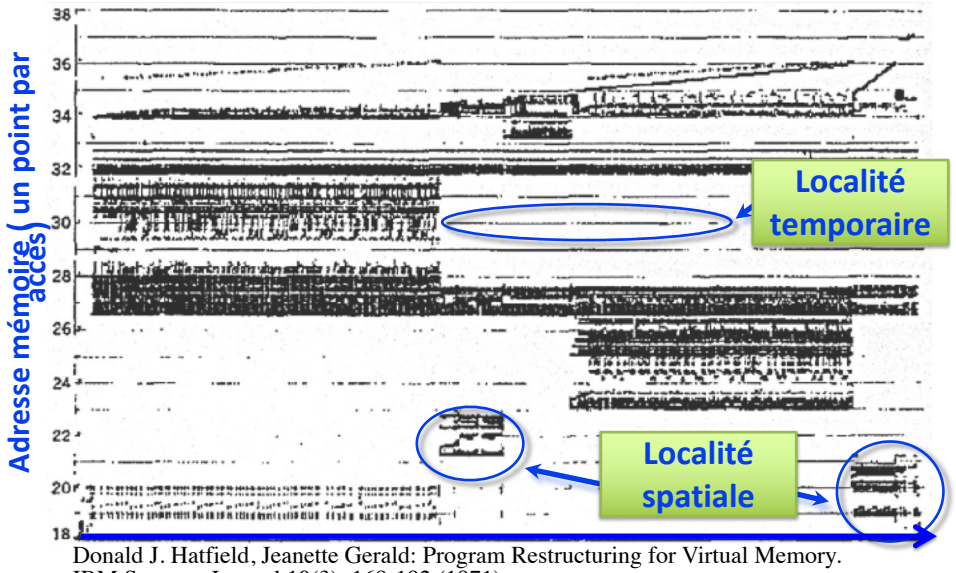
Principes de localité

- Observation: les références aux données et surtout aux instructions ne sont pas, d'habitude, indépendantes. Les programmes ont tendance à réutiliser les données et les instructions qu'ils ont utilisées récemment.

☐ Localité spatiale: les éléments dont les adresses sont proches les unes des autres auront tendance à être référencés dans un temps rapproché (p.ex. instructions, images).

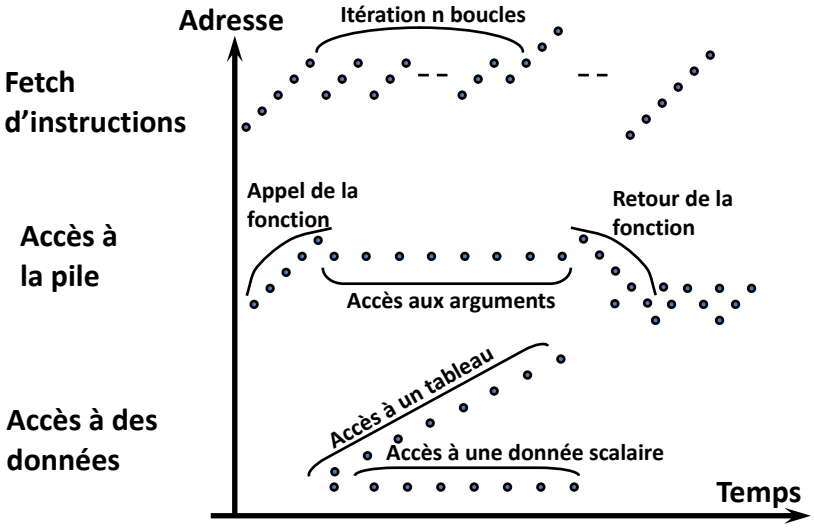
☐ Localité temporelle: les éléments auxquels on a eu accès récemment seront probablement accédés dans un futur proche (p.ex. boucles).

Cartographie des accès mémoire



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

Memory Reference Patterns



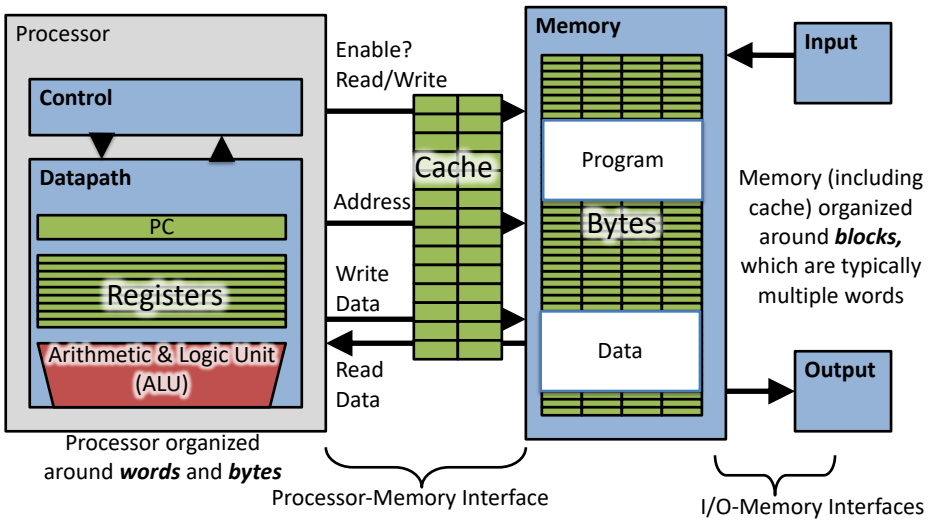
Accès Mémoire sans cache

- Instruction Load word : `lw t0,0(t1)`
- t1 contient $1022_{ten,}$ Mem[1022] = 99

1. Le processeur génére l'adresse 1022_{ten} vers la mémoire
2. Le mot contenu en mémoire à l'adress1022_{ten} est lu
3. La mémoire (interface) envoie la valeur 99 vers le processeur
4. Le processeur charge la valeur 99 dans le registre t0

10

Si on rajoute une mémoire cache…



11

12

Questions

À cause de la localité spatiale, les données et les instructions sont transférées de la mémoire principale au cache en petits blocs de 2-8 mots mémoire. Plusieurs questions s'imposent:

- Où peut-on placer un bloc dans le cache?
  - Placement de bloc
- Comment trouver un bloc s'il est présent dans le cache?
  - Identification de bloc
- Quel bloc doit être remplacé en cas d'échec?
  - Remplacement de bloc
- Qu'arrive-t-il lors d'une écriture?
  - Stratégie d'écriture

Accès Mémoire avec un cache

- Instruction Load word : `lw t0,0(t1)`
- t1 contient $1022_{ten,}$ Mem[1022] = 99
- Cache: Le processeur génére la requête d'accès à l'adresse 1022_{ten} à la mémoire cache
  1. Le contrôleur de cache vérifie si une copie de la donnée se trouvant en mémoire à l'adresse 1022_{ten} est présente dans le cache
     2a. Si la donnée est présente (**Hit**): le cache renvoie la valeur 99 au processeur
     2b. Si non présente (**Miss**): la mémoire principale est accédée à l'adresse 1022 to
        I. Le mot contenu en mémoire à l'adress1022_{ten} est lu
        II. La valeur lue est envoyée vers le cache
        III. Le cache enregistre cette nouvelle valeur
        IV. Le cache envoie la valeur (99) au processeur
  2. Le processeur charge la valeur 99 dans le registre t0

13

Les Tags

- Le contrôleur de cache a besoin d'un moyen de savoir si une donnée de la mémoire principale est présente dans le cache (hit ou miss)
- Lors d'un miss, on stocke l'adresse du bloc chargée dans la mémoire cache dans le contrôleur de cache
  - Ici 1022 est stocké dans le tag associé à la valeur 99

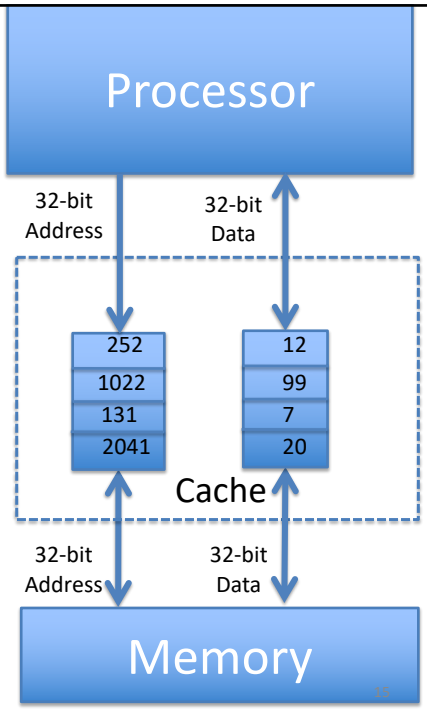
Tag	Data
252	12
1022	99
131	7
2041	20

Données en cache des précédents load/store

14

Mémoire cache de de 16 Octets avec des blocs de 4 octets

- 3 Opérations:
  1. Cache Hit
  2. Cache Miss
  3. Chargement du cache avec les données de la mémoire principale.
- Le cache a besoin des tags pour savoir s'il s'agit d'un hit ou d'un miss.
  - Les 4 tags sont comparés



15

Remplacement

- Imaginons que le processeur accède à l'adresse 511 qui contient la donnée 11.
- Aucun tag ne correspond, on doit donc supprimer une ligne du cache pour charger ce nouveau bloc.
  - Quel bloc remplacer?

Tag	Data
252	12
1022	99
131	7
2041	20

16

Remplacement

- Imaginons que le processeur accède à l'adresse 511 qui contient la donnée 11.
- Aucun tag ne correspond, on doit donc supprimer une ligne du cache pour charger ce nouveau bloc.
  - Quel bloc remplacer?
- On remplace la "victims" avec le nouveau bloc de l'adresse 511

Tag	Data
252	12
1022	99
511	11
2041	20

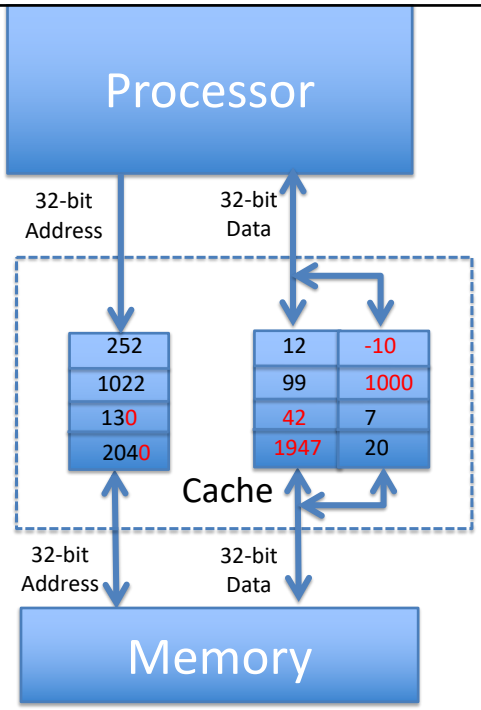
17

Gestion des Tags

- Les mots stockés an mémoire sont alignés donc l'adresse binaire d'un mot se termine toujours par 00_{two}
- Il n'est donc pas nécessaire de stocker les 2 derniers bits dans le tag
  - Cela simplifie le contrôleur et nécessite moins de capacité de stockage.

Cache de 32 octets avec des blocs de 8 octets

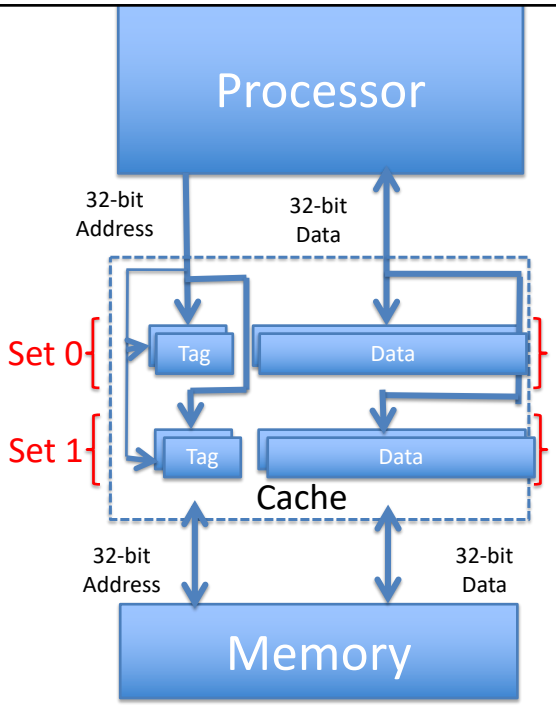
- Les 3 derniers bits de l'adresse du bloc valent toujours 000_{two}
- On a un hoit si on accede un des deux mots du bloc



19

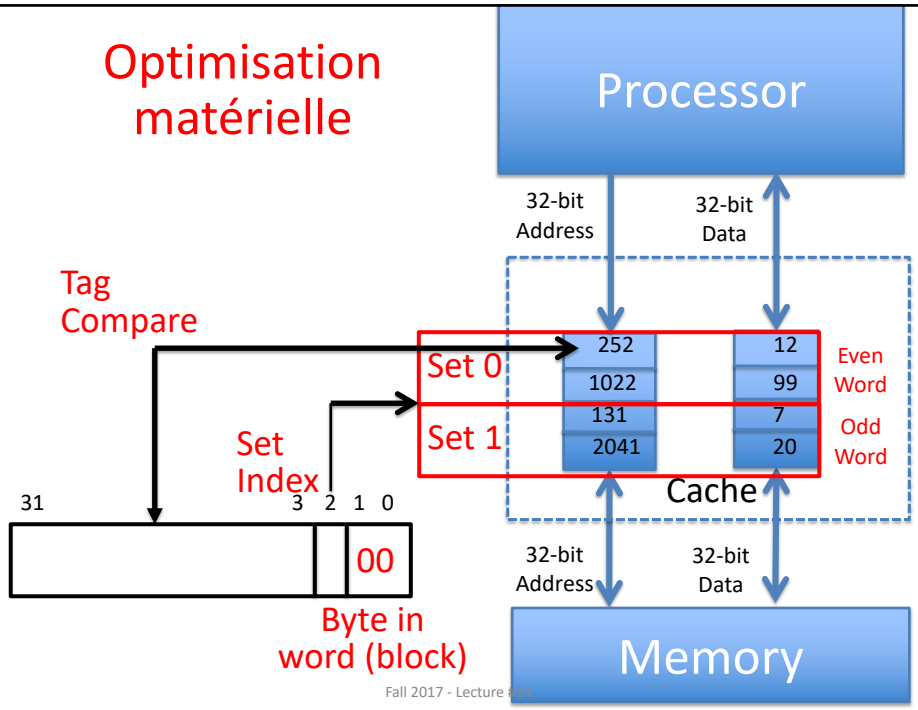
Optimisation matérielle

- Il faut comparer tous les tags du cache avec l'adresse
- Les comparateurs ont un cout non négligeable.
- Une optimisation: utilisation de deux ensembles "sets" de données pour limiter les comparaisons.
- On utilise un bit de l'adresse pour sélectionner un ensemble.
- On compare l'adresse uniquement avec les tags de l'ensemble
- On peut généraliser cette approche à plus d'ensembles.



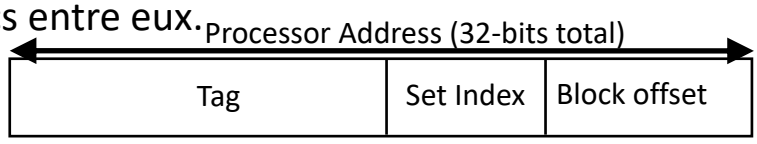
20

Optimisation matérielle



Quels champs de l'adresse utilise le controleur de cache?

- Block Offset: adresse de l'octet dans le bloc
- Set Index: indique quel ensemble (set)
- Tag: les bits restant de l'adresse qui permettent de différencier les blocs entre eux.



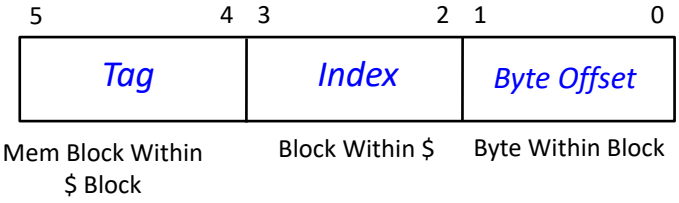
- Taille de l'index= $\log_2$(number of sets)
- Taille du tag= Address size – Size of Index
  – $\log_2$(number of bytes/block)

Qu'est ce qui limite le nombre d'ensembles?

- Pour un nombre total de blocs, on réduit le nombre de comparaisons si on a plus de deux ensembles.
- Une limite: Si on a autant d'ensemble que de bloc=> seulement un bloc par ensemble, une seule comparaison à effectuer!
- C'est ce qu'on appelle les caches à correspondance directe

23

Cache à correspondace directe: Mémoire avec un bus d'adresse de 6 bits



- Taille des blocs 4 octets
- Les blocs de la mémoire cache et de la mémoire principale ont toujours la même taille
- # blocs en mémoire>> # blocs en cache
  - 16 blocs en mémoire = 16 words = 64 bytes => 6 bits pour addresser tous les bytes
  - 4 blocs en cache, 4 bytes (1 word) par bloc
  - 4 blocs de la mémoire peuvent être contenu dans la cache
- Dans quel ensemble peut se trouver un bloc de la mémoire en cache: c'est le role de l'index
- Quel bloc de la mémoire principale est dans le cache : information donnée par le tag

24

Bit de Validité

- Lorsqu'on démarre un nouveau programme, le cache ne contient pas des informations valides pour ce programme.
- On a besoin d'indiquer cette information
- C'est le rôle du bit de validité
  - 0 => cache miss, même si on a le bon tag
  - 1 => cache hit, si on le bon tag

25

Les différentes architectures de mémoire cache

26