Architecture des Ordinateurs et Systèmes d'Exploitation

Cours n°8

Mémoire(s) : caractéristiques
Hiérarchie des mémoires
Mémoire cache

INSA ROUEN
Ph. Leray

asi Architecture des Systèmes d'Information
3ème année

Caractéristiques principales

- **Méthodes d 'accès :**
  - **séquentiel (temps d 'accès linéaire)** — **bande**
  - **direct (temps d 'accès constant)** — **mémoire principale**
  - **mixte (accès direct au voisinage de la donnée + parcours séquentiel)** — **disque**
  - **associatif (accès à la donnée en recherchant une clé dans une table, temps constant)** — **cache**

- **Supports physiques**
  - **semi-conducteur**
  - **magnétique**
  - **optique**

- **Caractéristiques**
  - **volatile / non volatile**
  - **effaçable / non effaçable**

Caractéristiques (2)

- **Capacité de la mémoire :**
  - **nombre de mots**
  - **taille du mot (généralement 1 octet)**
- **Unité de transfert : mot, bloc, page, fichier, …**
- **Performances :**
  - **Temps d 'accès**
  - **Temps de cycle (= entre 2 accès)**
  - **Débit de transfert (= 1 / Temps Cycle)**

Quelques exemples :

- **RAM (Random Access Memory / Mémoire Vive)**
  - **accès direct (temps d 'accès = 10 ns (SRAM) - 70 ns (DRAM))**
  - **volatile**
  - **accès : mode normal / mode page**

- **ROM (Read Only Memory / Mémoire Morte)**
  - **accès direct**
  - **non volatile : information figée lors de la fabrication**
  - **utilisation : microprogrammes + programmes systèmes de base (BIOS)**

- **PROM (Programmable ROM) : ROM enregistrable 1 fois par l 'utilisateur**
- **EPROM (Erasable PROM) : PROM effaçable (par rayonnement UV)**
- **EEPROM (Electricaly EPROM) : PROM effaçable (signaux électriques)**
- **FLASH EEPROM : EEPROM effaçable par bloc**

« Mémoire idéale »

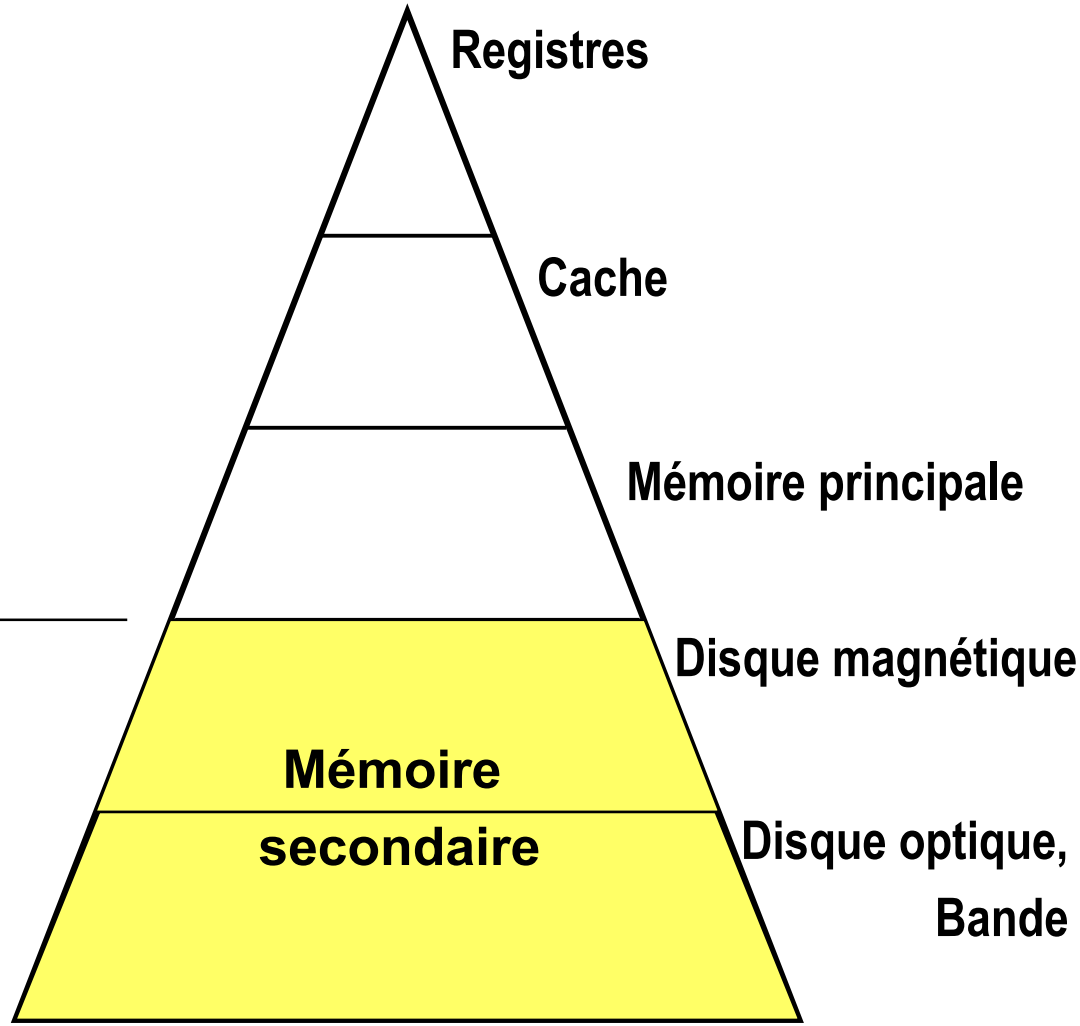
- **Mémoire idéale = grande capacité, rapide, pas chère.**

- **La réalité :**
  - **les mémoires de grande capacité sont lentes**
  - **les mémoires rapides sont chères**
  - **Ex:**
    - » **Processeur avec une horloge de 200 MHz (cycle = 5 ns)**
    - » **DRAM : temps de cycle de 70 ns**
    - → **1 cycle d 'accès à la mémoire = 14 cycles processeur !**

- **Comment faire ?**
  - **Faire des accès parallèles à la mémoire**
  - **Utiliser une hiérarchie de mémoires**

Hiérarchie mémoire

Avantages	Inconvénients
Rapidité	Volatile
Accès direct	Faible capacité
Faible taille	Chère (prix/bit)
Peu chère	Lenteur
Non volatile	Volumineuse
Grande capacité	Accès séquentiel / mixte



Hiérarchie mémoire: fonctionnement

- **Une unité de transfert d 'informations passe seulement entre 2 niveaux de mémoire adjacents :**
  - **Niveau supérieur = le plus proche du processeur**
    (plus rapide mais plus cher et faible capacité)
  - **Niveau inférieur = le plus loin du processeur**
    (moins rapide, peu cher mais de forte capacité)

- **Calcul des performances :**
  - **taux de succès = % des accès trouvant l 'information dans le niveau supérieur**
  - **temps de succès = temps d accès à l 'information dans le niveau supérieur**
  - **taux d'échec = 1 - taux de succès**
  - **Pénalité d'echec = temps de remplacement d 'un bloc dans le niveau supérieur**

→ Bonne gestion de la mémoire : Temps succès << Pénalité d'échec

Hiérarchie mémoire: fonctionnement (2)

Comment gérer correctement cette hiérarchie ?

- **Principe de localité :**

  à tout instant, un programme accède à une partie relativement petite de son espace d 'adressage

- **Localité temporelle :**

  si un mot est accédé, on a de fortes chances d'y accéder à nouveau dans les instants qui suivent

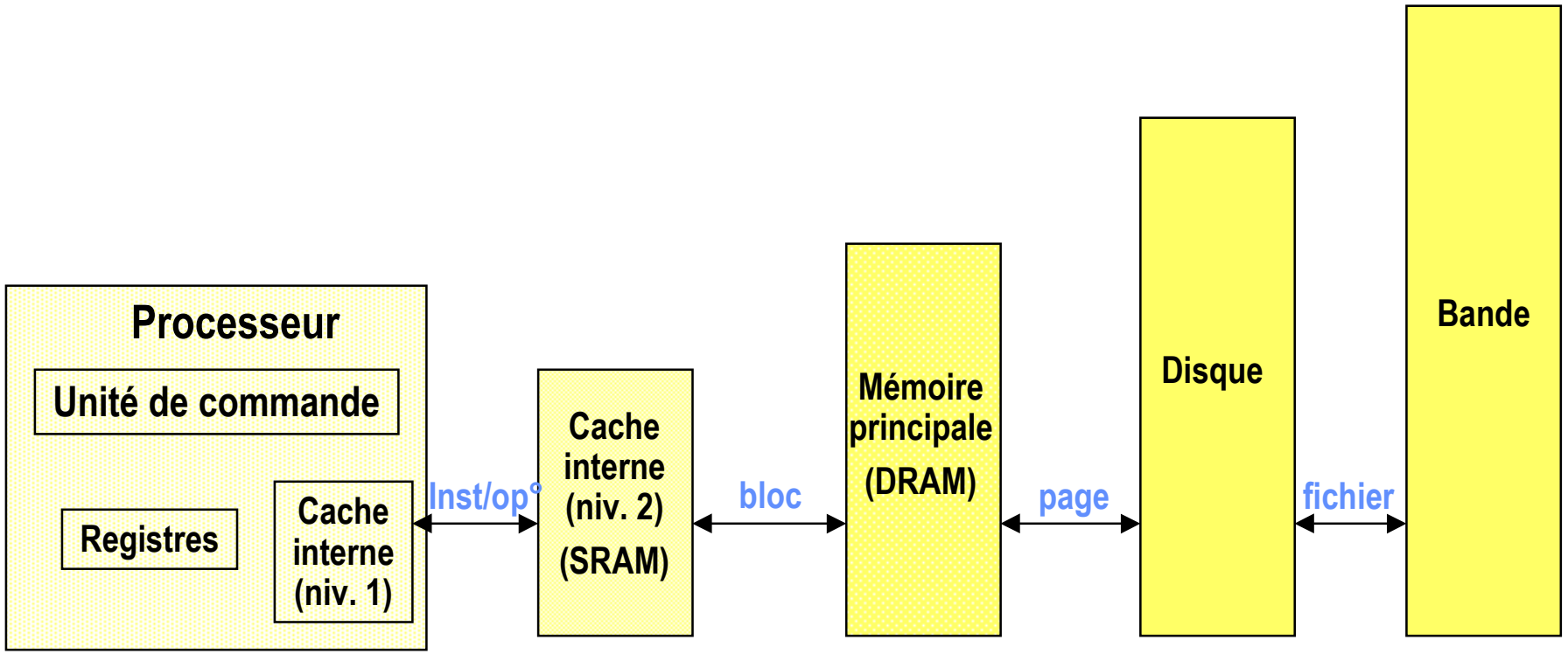
  → garder les mots les plus fréquemment accédés

- **Localité spatiale :**

  si un mot est accédé, on a de fortes chances d'accéder à un mot ayant une adresse voisine

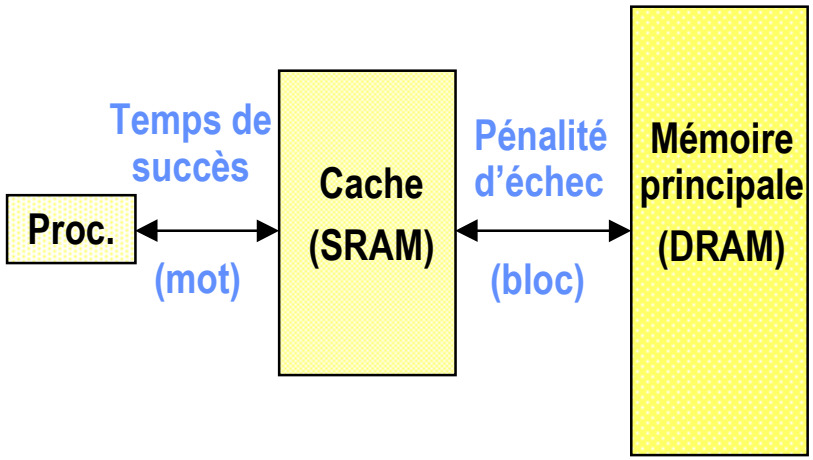
  → garder des mots contigus (par bloc)

Ex. de Hiérarchie mémoire



Accès (ns) : 1	10	10^2	10^7	10^{10}
Taille : 100 oct.	Ko	Mo	Go	To

Mémoire cache : performances



Temps d 'accès moyen = Temps de succès + Taux d'échec * Pénalité d'échec

Ex: T accès SRAM = 10 ns

T succès cache = 90%

T accès DRAM = 70 ns

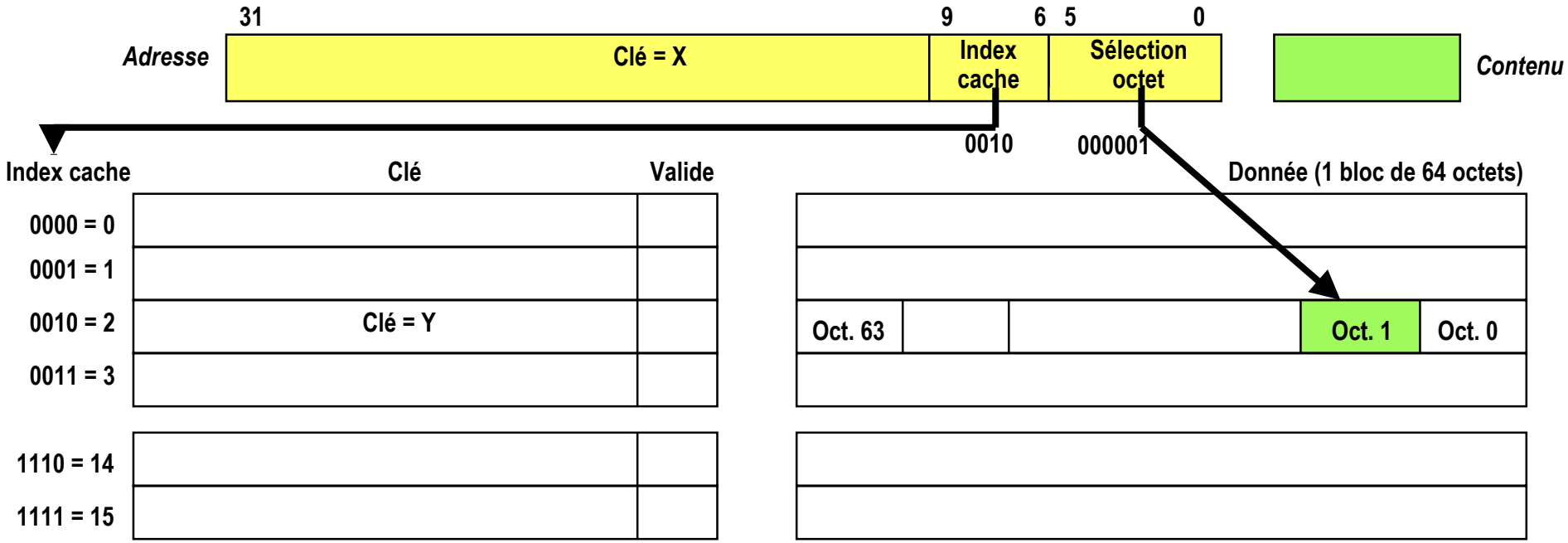
T accès DRAM en mode page = 70 + 20 + 20 + … + 20 ns

= 70 + 7*20 ns si Bloc = 8 mots mémoire

T accès moyen = 10 + 10% * (70 + 7*20) = 31 ns

Mémoire cache : structure (1)

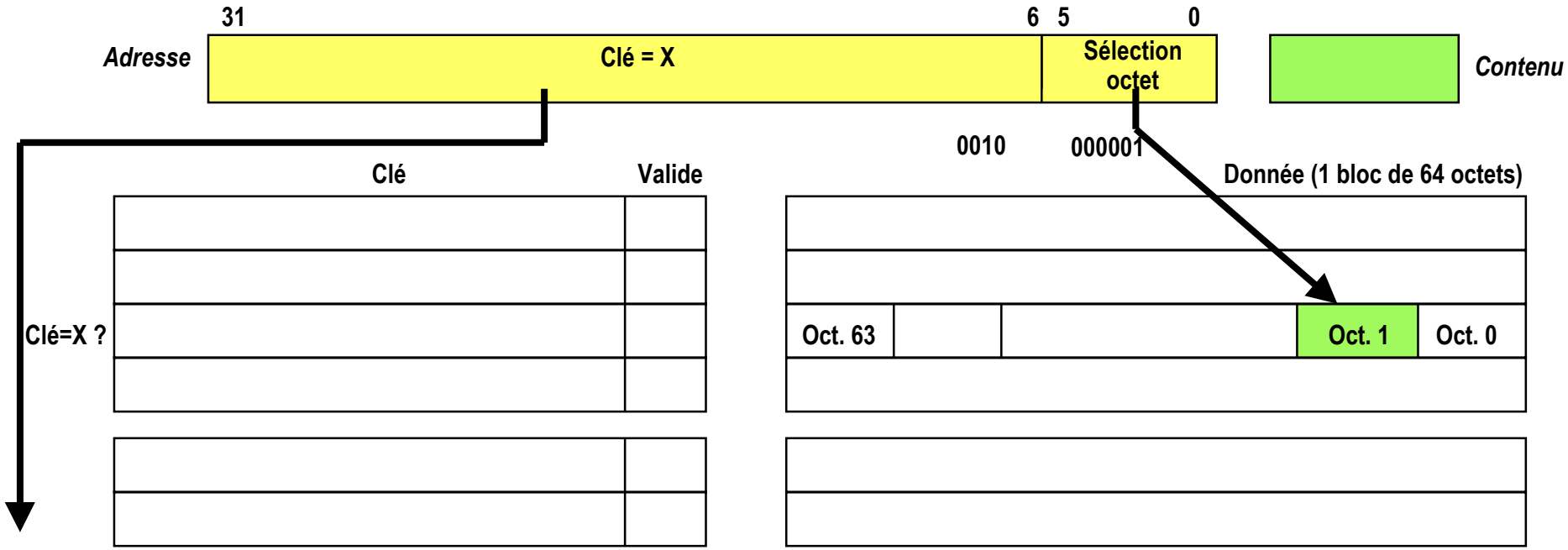
- **Structure n°1 : cache direct**
  - chaque bloc est rangé selon ses bits d'adresse de poids faible
  - Ex: cache de 1 Ko = 16 blocs de 64 octets



  - Lecture du cache: si X=Y et Valide=1 : Succès ! (Sinon Echec)
  - Avantage: décision rapide (on cherche directement à la ligne donnée par l 'index)
  - Inconvénient: *ping-pong* (impossible d'avoir dans le cache 2 blocs de même index)

Mémoire cache : structure (2)

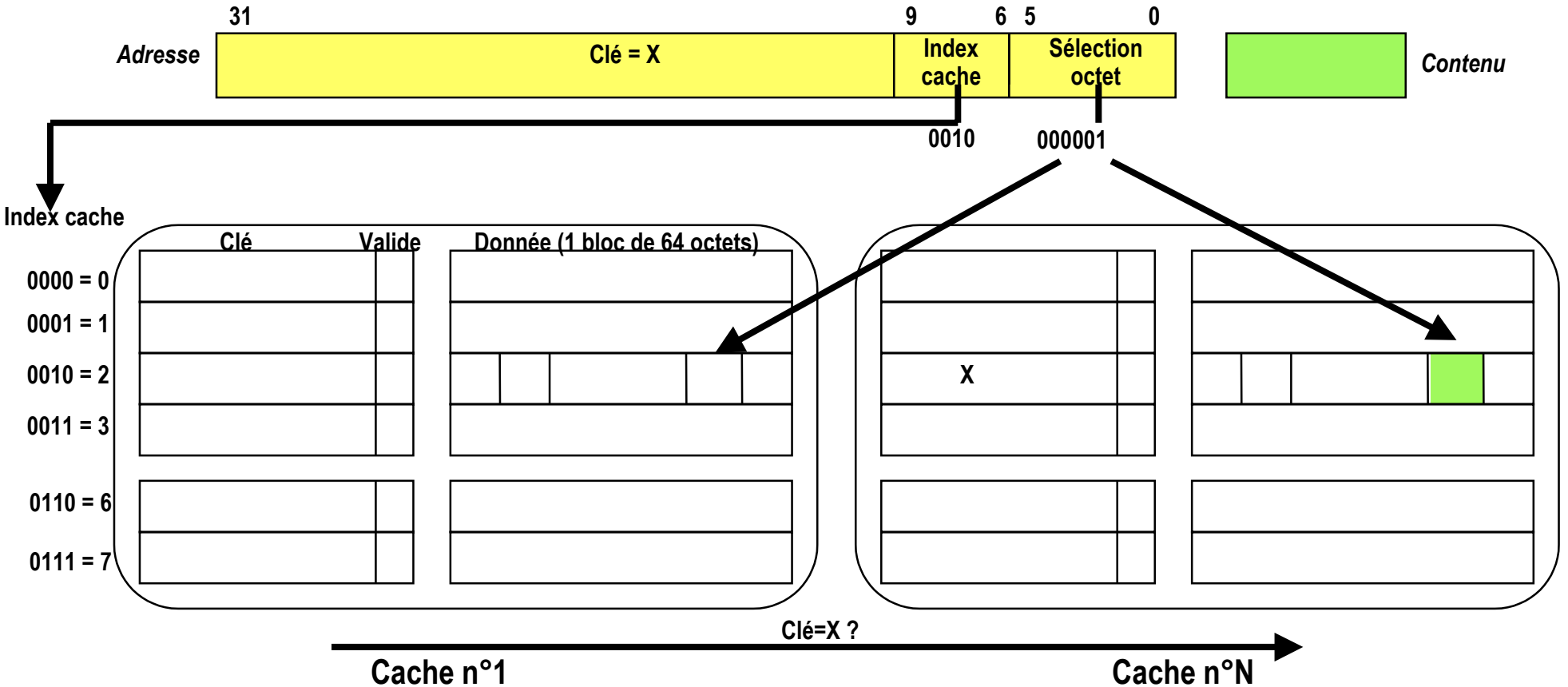
- **Structure n°2 : cache associatif**
  - chaque bloc est rangé n'importe où
  - Ex: cache de 1 Ko = 16 blocs de 64 octets



  - Lecture du cache: si X est dans le cache et Valide=1 : Succès ! (Sinon Echec)
  - Avantage: plus de *ping-pong*
  - Inconvénient: décision plus lente (il faut parcourir toutes les clés du cache)

Mémoire cache : structure (3)

- **Structure n°3 : cache associatif à N voies**
  - **= N caches directs fonctionnant en parallèle**
  - **Ex: cache de 1 Ko = 16 blocs de 64 octets**



Mémoire cache : écriture

- **Cache direct**

  Chaque bloc peut être placé dans un seul emplacement du cache

  ⇒ pas besoin de décision en cas de nouvelle écriture

  ⇒ (le nouveau bloc remplace l'ancien)

- **Cache associatif**

  Chaque bloc peut être placé n'importe où dans le cache

  ⇒ décision à prendre en cas de nouvelle écriture

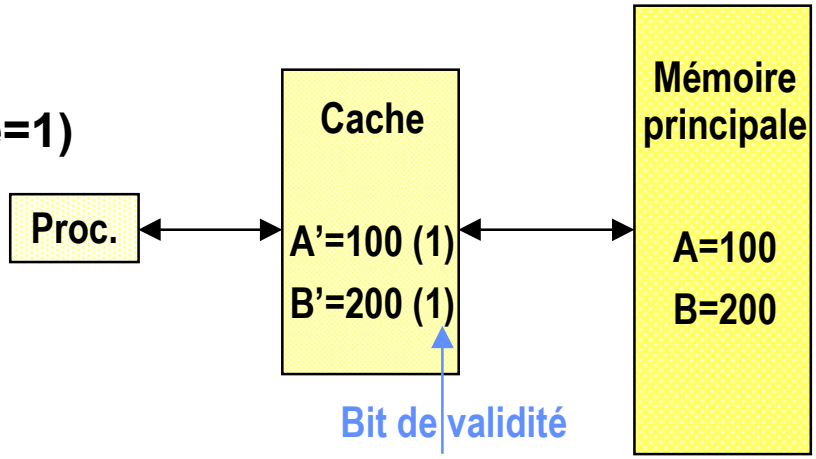
  ⇒ faut-il jeter un ancien bloc ? Si oui, lequel ?

  Politique de remplacement :

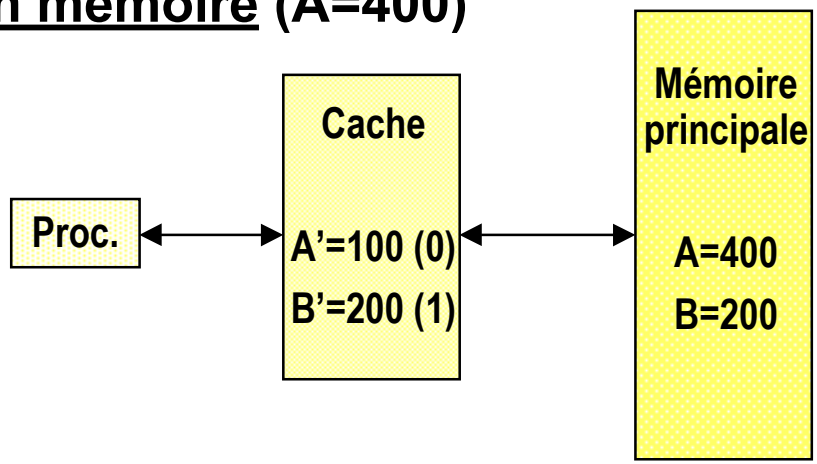
  - **aléatoire**
  - **LRU (Least Recently Used) : le gestionnaire du cache conserve l 'historique des accès et remplace le bloc qui a été utilisé le moins récemment**

Cohérence Mém. cache/ Mémoire (1)

- **Situation cohérente**
  (A=A' et valide=1 ; B=B' et valide=1)

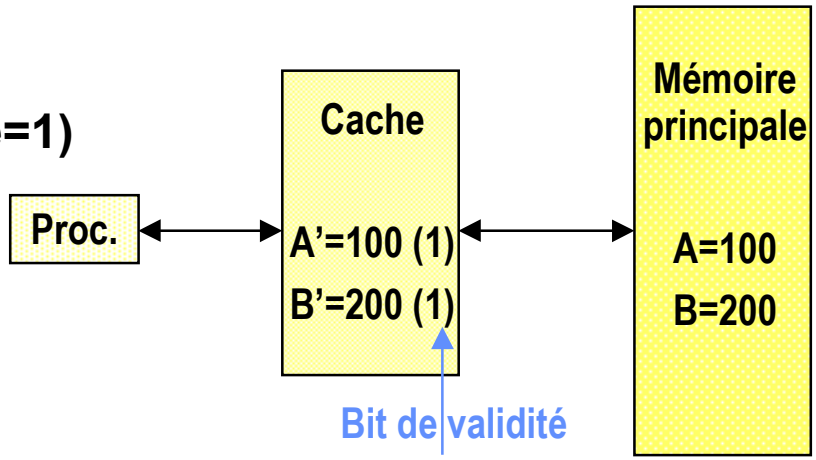


- **Une E/S modifie une valeur <u>en mémoire</u> (A=400)**
  cache et mémoire incohérents

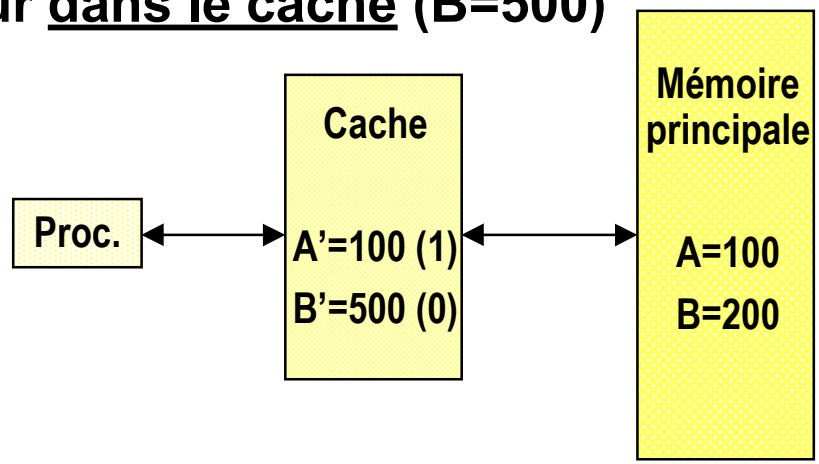


Cohérence Mém. cache/ Mémoire (2)

- **Situation cohérente**
  (A=A' et valide=1 ; B=B' et valide=1)



- **Le processeur écrit une valeur dans le cache (B=500)**
  cache et mémoire incohérents



Cohérence Mém. cache/ Mémoire (3)

- **Ecriture en cas de succès cache (la variable est déjà dans le cache)**

  2 politiques d'écriture possibles

  - **écriture immédiate :**
    - » **on met à jour le cache et la mémoire en même temps**
    - ⇒ **pb de «lenteur» d 'accès à la mémoire**

  - **écriture différée :**
    - » **on écrit seulement dans le cache**
    - » **la mise à jour de la mémoire se fait seulement quand le bloc est supprimé du cache (remplacé par un autre)**
    - ⇒ **besoin d 'un bit supplémentaire (*Modifié*)**
    - ⇒ **avantage = réduction du trafic vers la mémoire**
    - ⇒ **inconvénient = gestion du cache plus complexe**

Cohérence Mém. cache/ Mémoire (4)

- **Ecriture en cas d'échec cache (la variable n'est pas dans le cache)**

  Question = doit-on amener le bloc correspondant dans le cache ?

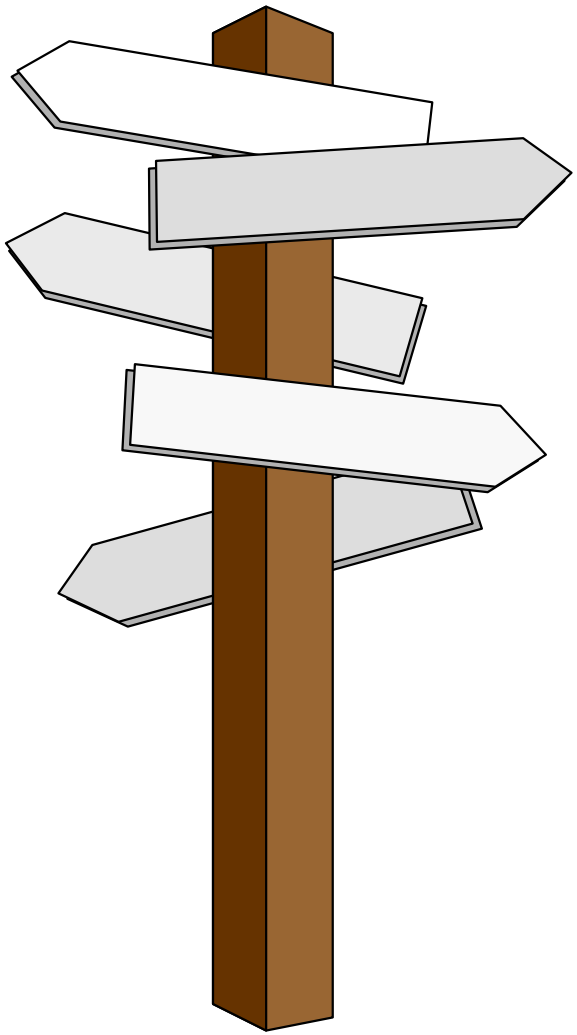
  - **OUI = écriture allouée :**
    - » **un nouvel accès à cet bloc sera un succès**
    - ⇒ **avantage = taux d'échec réduit / marche bien avec écriture différée**
    - ⇒ **inconvénient = gestion du cache plus complexe**

  - **NON = écriture non allouée :**
    - » **un nouvel accès à cet bloc sera un échec**
    - ⇒ **avantage = gestion simple / marche bien avec écriture directe**
    - ⇒ **inconvénient = taux d'échec élevé**

Mémoire cache : Conclusion

- **Différents paramètres :**
  - **taille du cache**
  - **taille du bloc**
  - **structure (direct / associatif)**
  - **politique de remplacement**
  - **écriture immédiate / directe (en cas de succès)**
  - **allocation / non allocation (en cas d'échec)**

- **Types de caches (non abordé ici)**
  - **cache mixte :**
    - » **un cache pour le code et les données**
    - » **ex: cache niv. 2 (externe)**
  - **cache séparé :**
    - » **un cache pour les instructions et un pour les données**
    - » **ex: cache niv.1 (interne)**



Références

Hiérarchie de la mémoire - Mémoire cache :

- Architecture de l'Ordinateur - A. Tanenbaum (InterEditions)
- Architecture des Systèmes d'Exploitation - M. Griffiths & M. Vayssade (Hermès)
- Architecture des Ordinateurs: une approche quantitative - J.L. Hennessy & D.A. Patterson (Int. Thomson Publishing)
- Technologie des Ordinateurs et des Réseaux - P.A. Goupille (Dunod)
- Architecture et Technologie des Ordinateurs - P. Zanella & Y. Ligier (Dunod)

A suivre :

- TD n° 8 : Cache direct / Cache associatif
- Cours n°12 : allocation de la mémoire - mémoire virtuelle