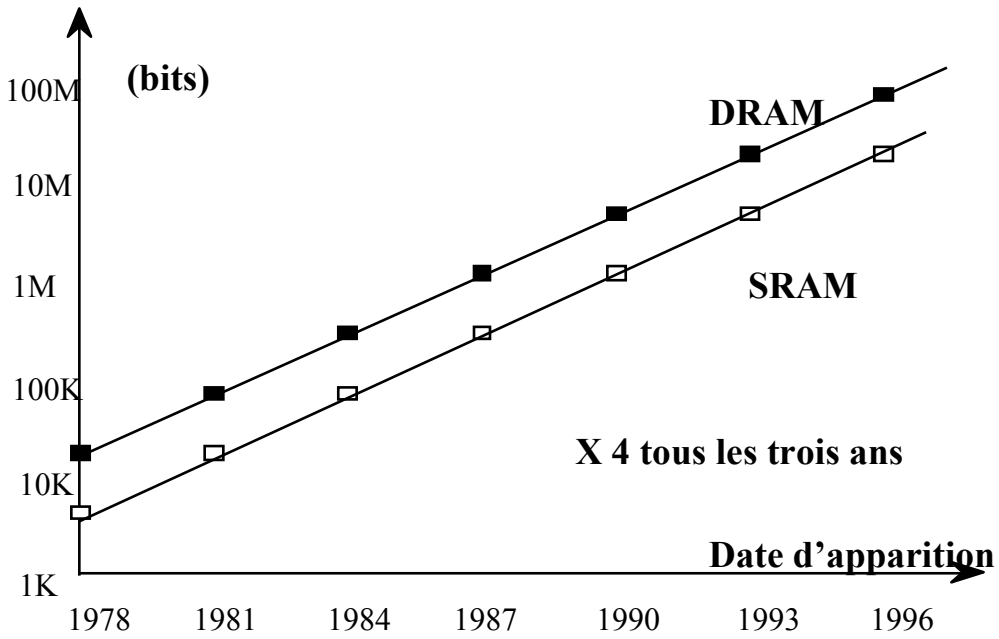
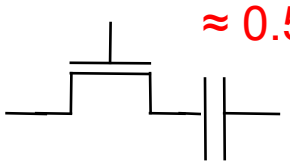
Architectures des ordinateurs
Caches et mémoire virtuelle

Daniel Etiemble
de@lri.fr

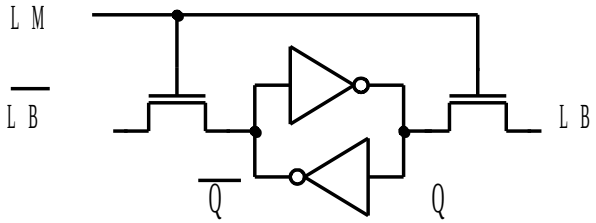
Capacité mémoire



Surface SRAM/DRAM = 4
Coût bit SRAM/DRAM = 10

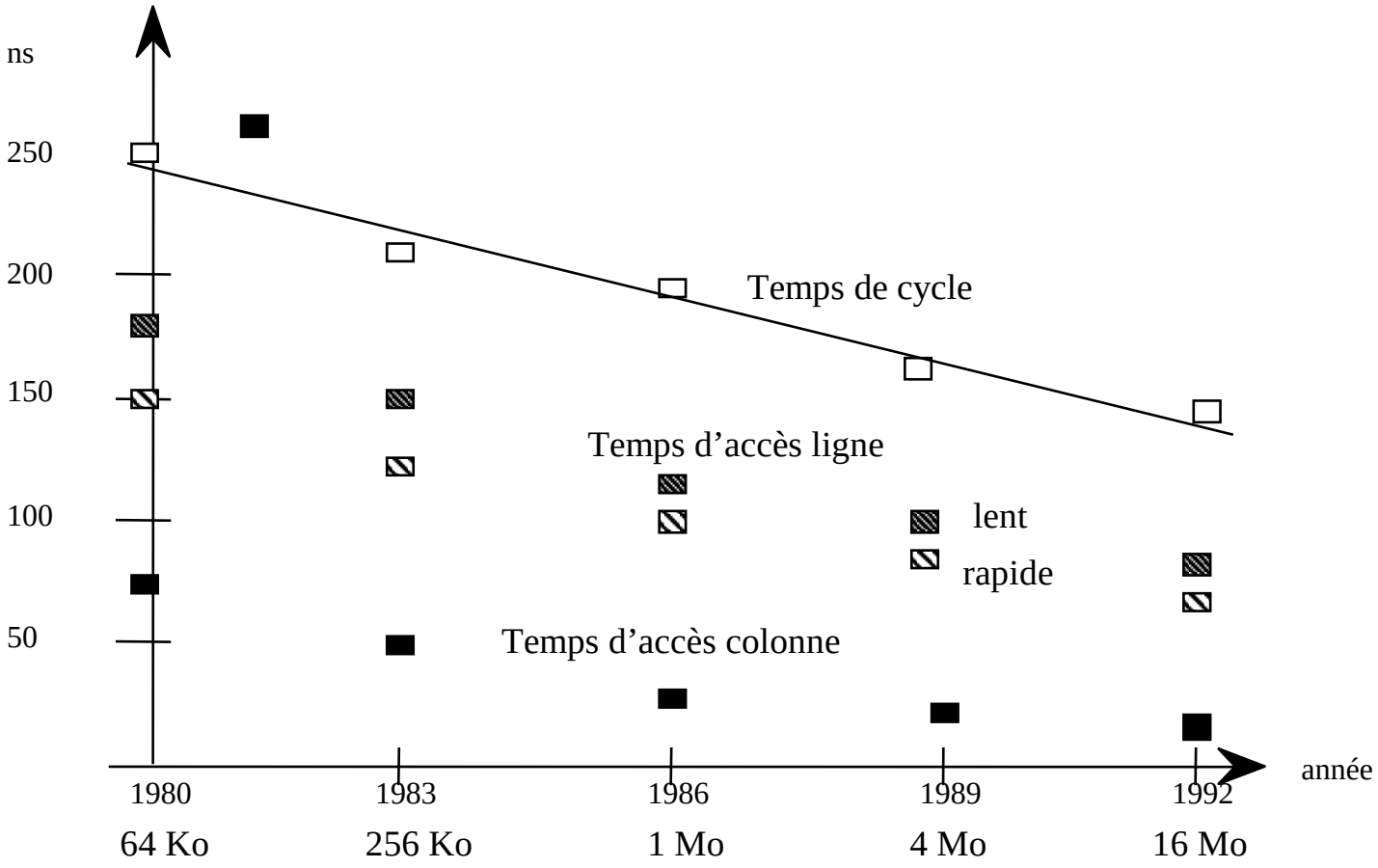


Point mémoire DRAM
(1,5 transistors)

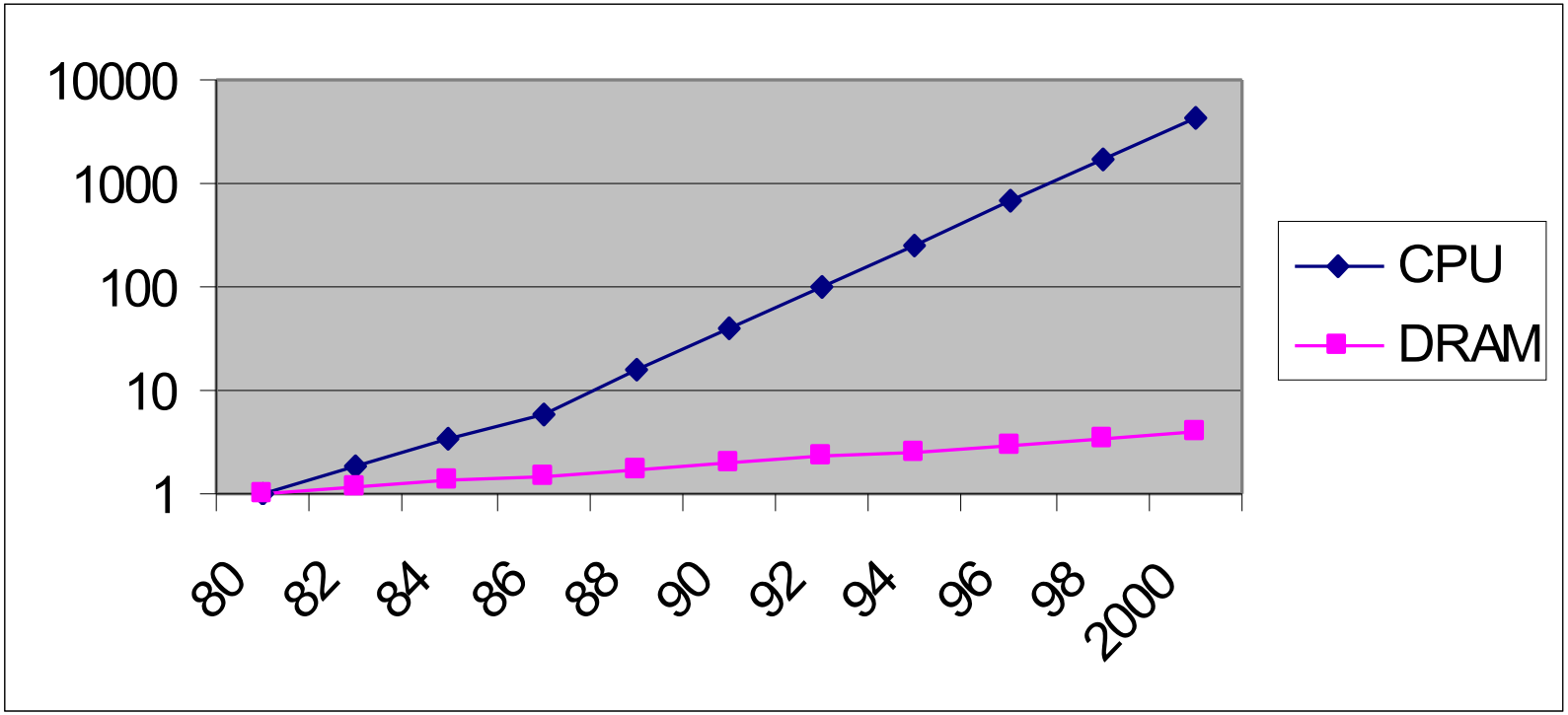


Point mémoire SRAM
(6 transistors)

Caractéristiques temporelles des DRAM

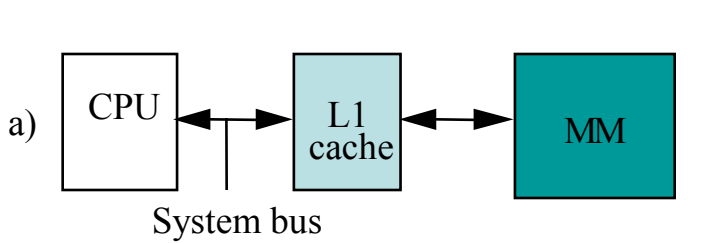


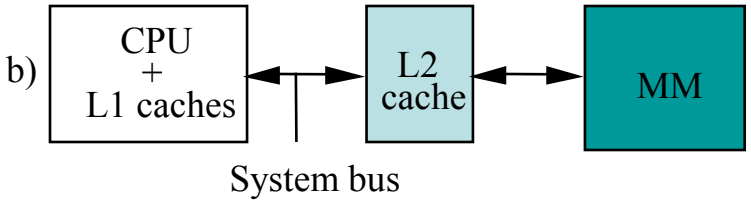
Le “gap” des performances mémoire



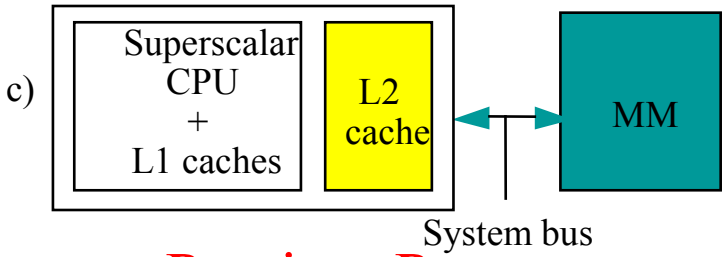
Croissance besoins CPU : 50 à 60%/an
Décroissance latence DRAM : 8 à 10%/an

Evolution des hiérarchies mémoire (1985-2000)

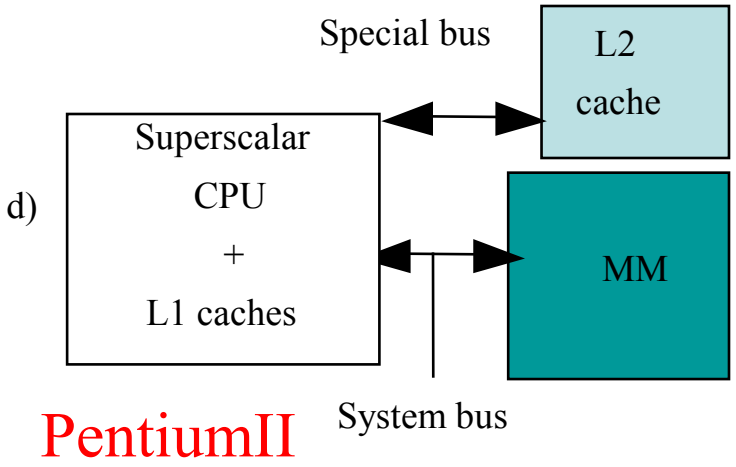




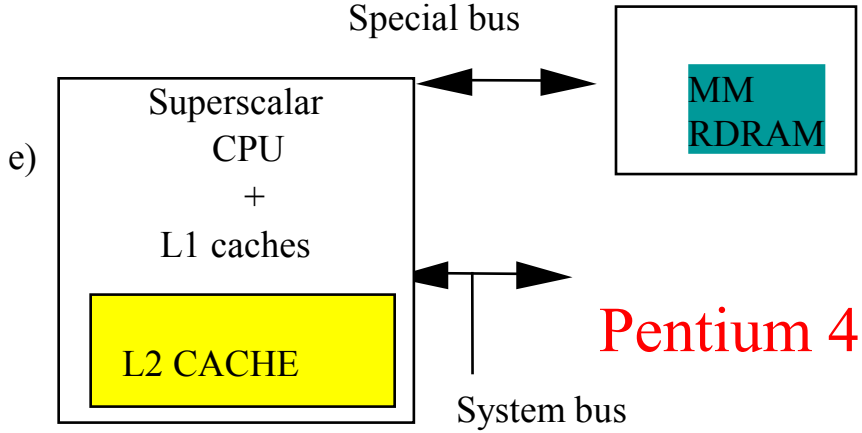
Pentium



Pentium Pro



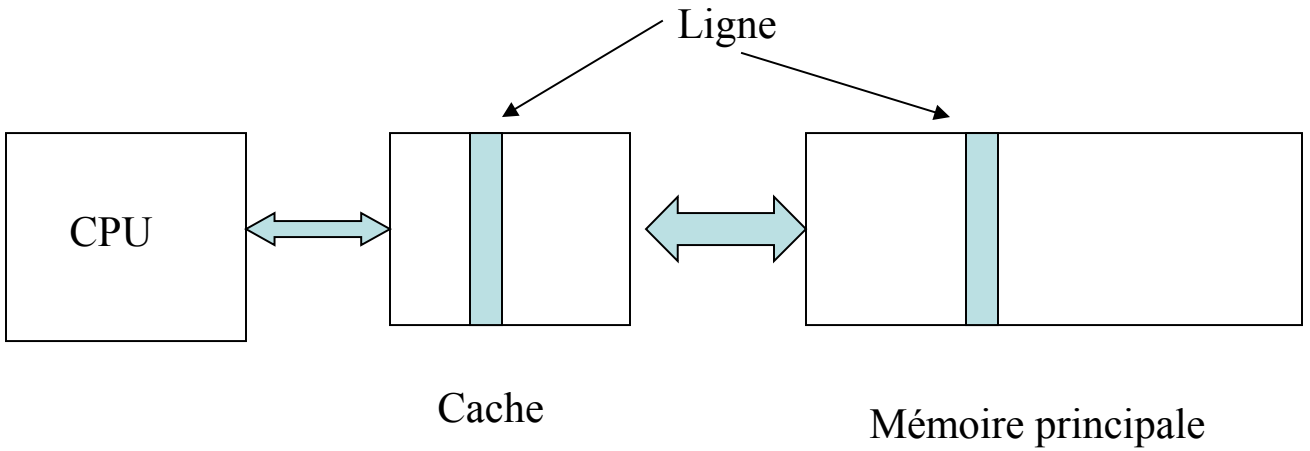
PentiumII



Pentium 4

Principe des caches

- Fondés sur le principe de localité
- Mémoires de taille et vitesse différentes
- Organisation
  - Découpage en lignes (blocs)
  - Mécanisme de correspondance
    - Placement des lignes dans le cache
    - Détection succès ou échec
  - Gestion de la cohérence



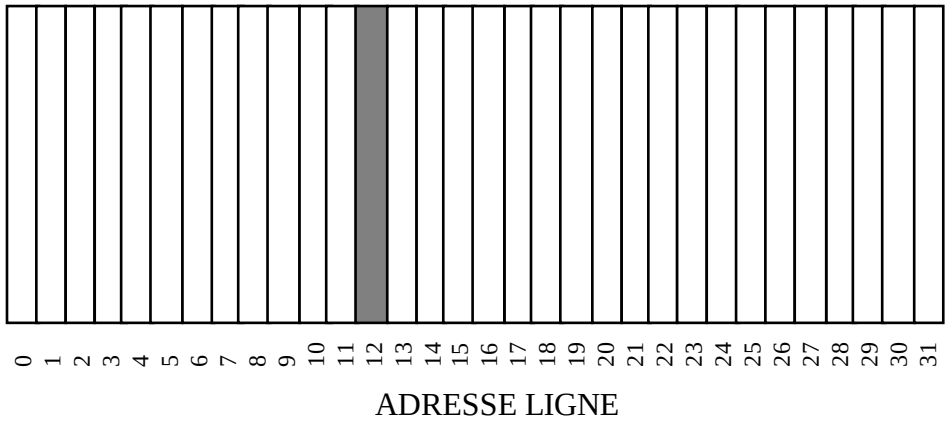
Principe de localité

- Localité spatiale
  - Si on accède à une case mémoire, on accédera à une case proche
- Localité temporelle
  - si on accède à une case mémoire, on y accédera probablement très bientôt (ou dans très longtemps ou jamais !)

Mécanismes de correspondance

OU PLACER UNE LIGNE DE LA MP DANS LE CACHE ?

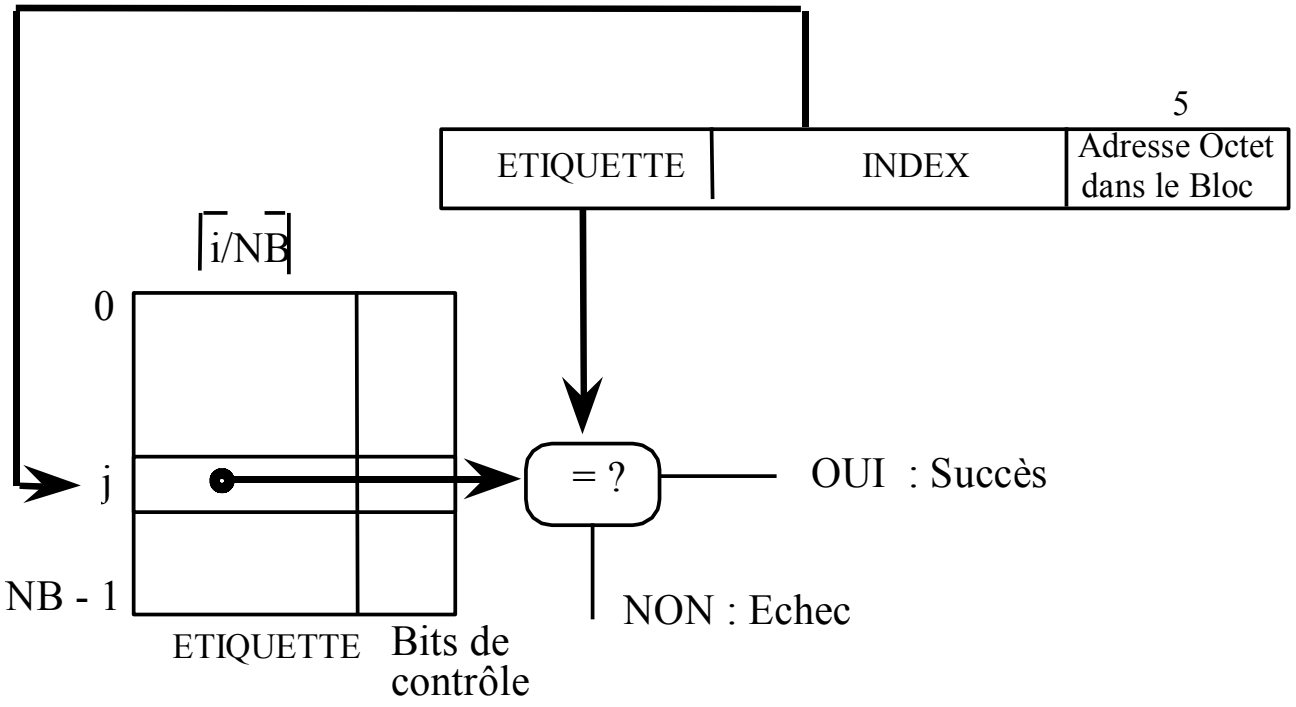
Mémoire principale



Correspondance directe Totalement associatif Associatif deux voies

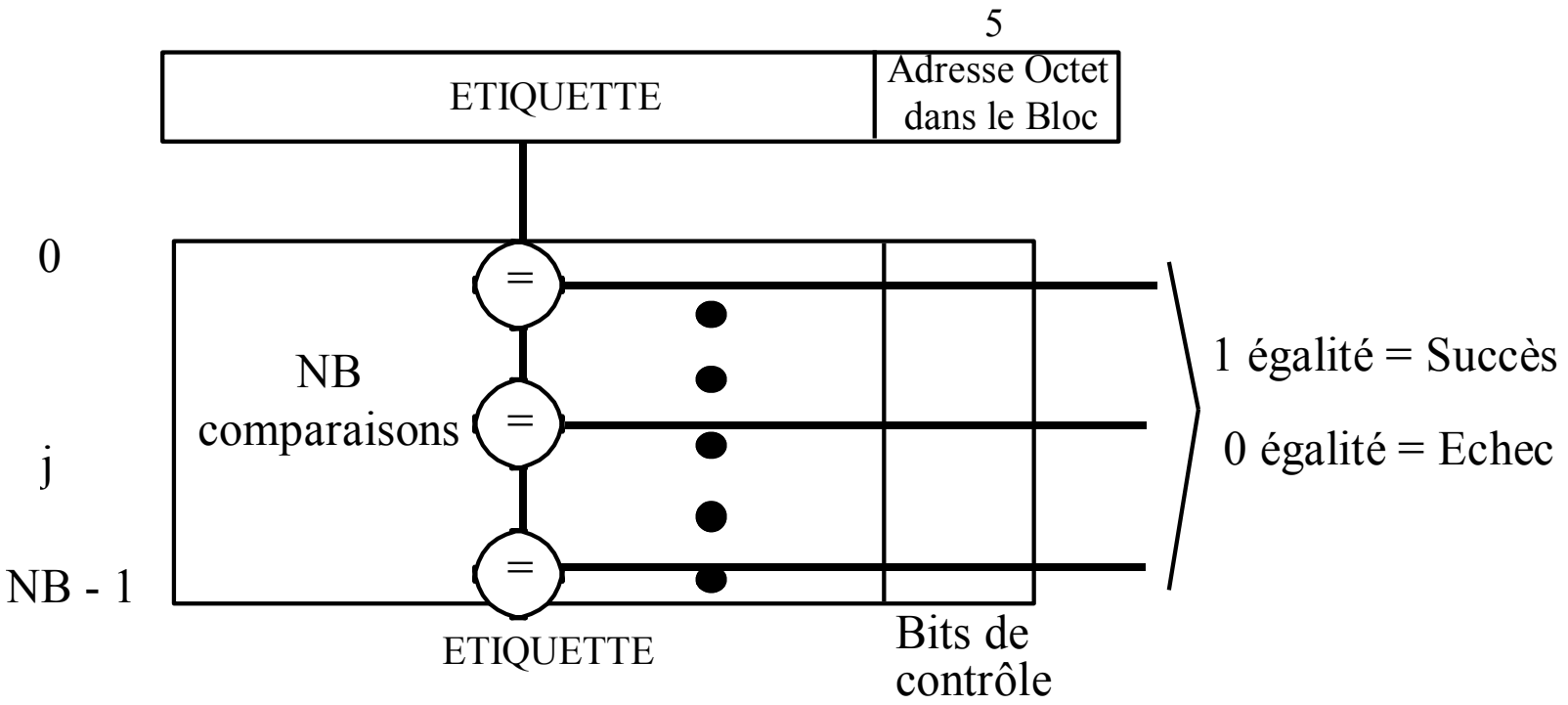
Correspondance directe

J (cache) = i (MP) mod NB, avec NB = nombre de lignes du cache



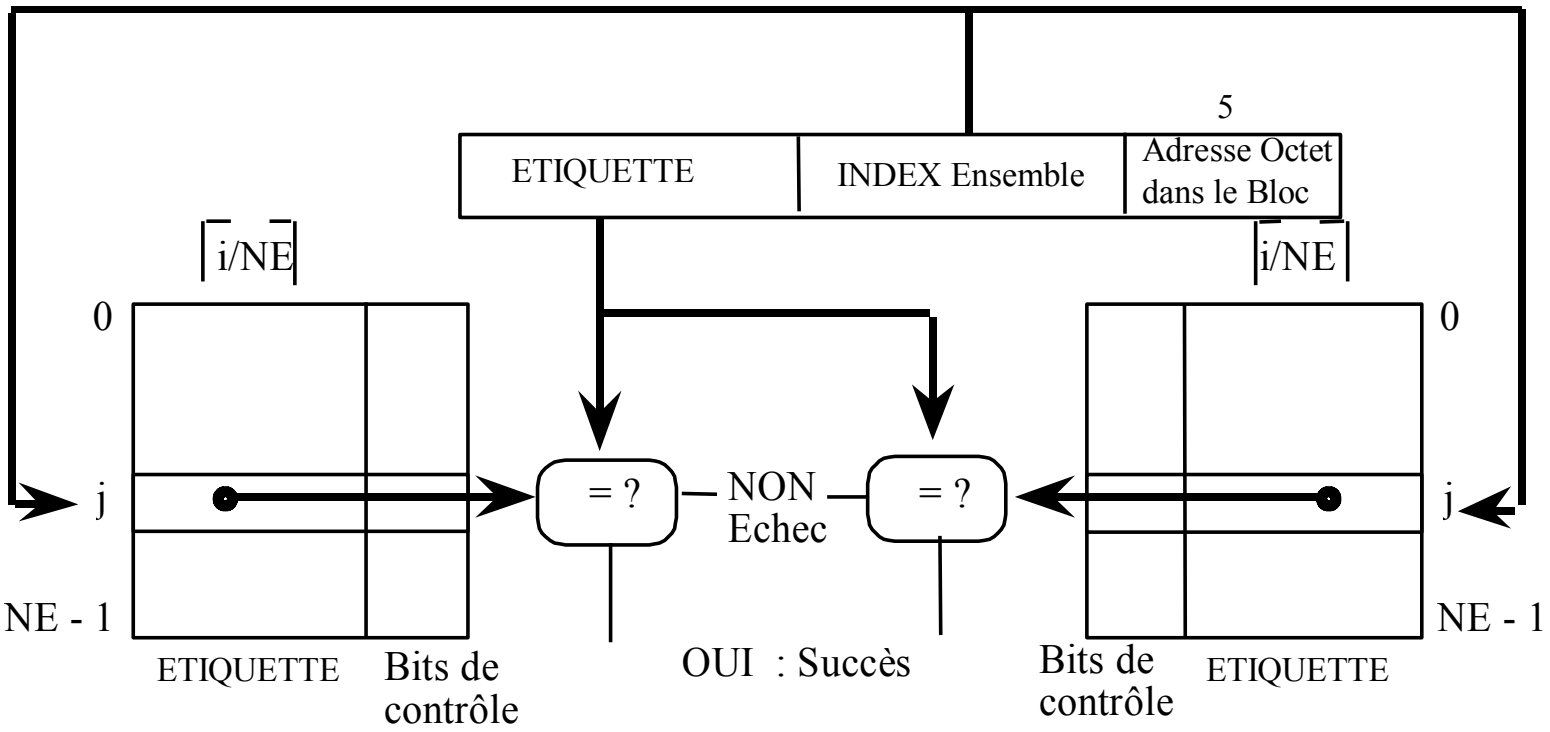
Associativité totale

Ligne i(MP) dans n'importe quelle ligne j du cache.
Etiquette = i, et NB comparateurs pour détecter succès ou échec



Associativité par ensemble

N (2, 4, 8) lignes par ensemble. Correspondance directe pour les ensembles et associativité à l'intérieur d'un ensemble.
N comparateurs.

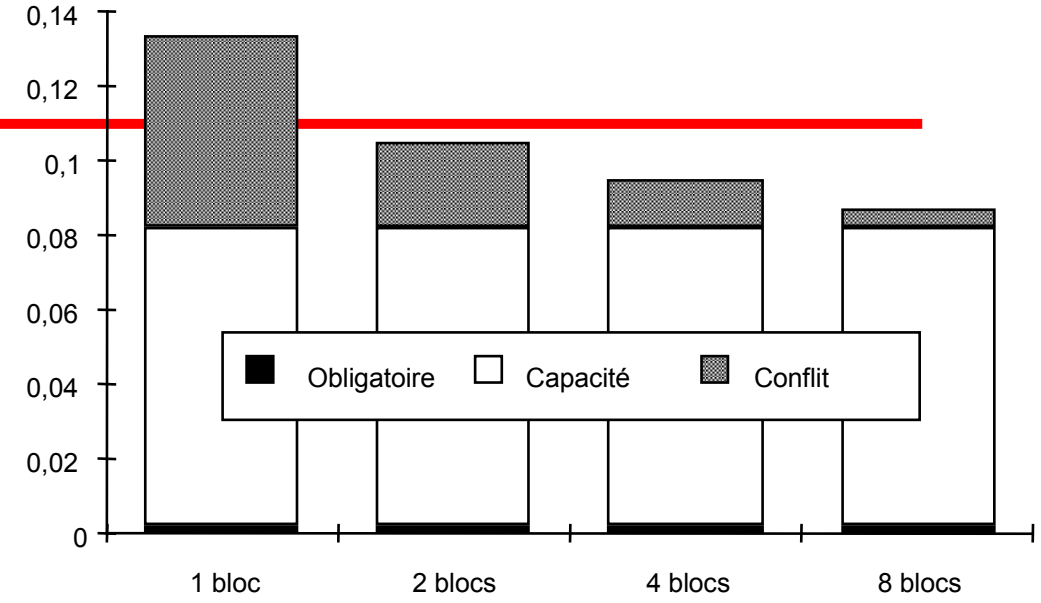


Les causes d'échec cache

- Échecs de démarrage
  - Premiers accès à une donnée ou une instruction : elle est en mémoire principale.
- Échecs de capacité
  - Le cache est trop "petit" par rapport aux besoins du programme (instructions et/ou données)
- Échecs de conflit
  - Le mécanisme de correspondance utilisé (correspondance directe ou associativité par ensemble remplace des lignes déjà présentes dans le cache (alors que d'autres lignes peuvent être non utilisées)

Les 3C

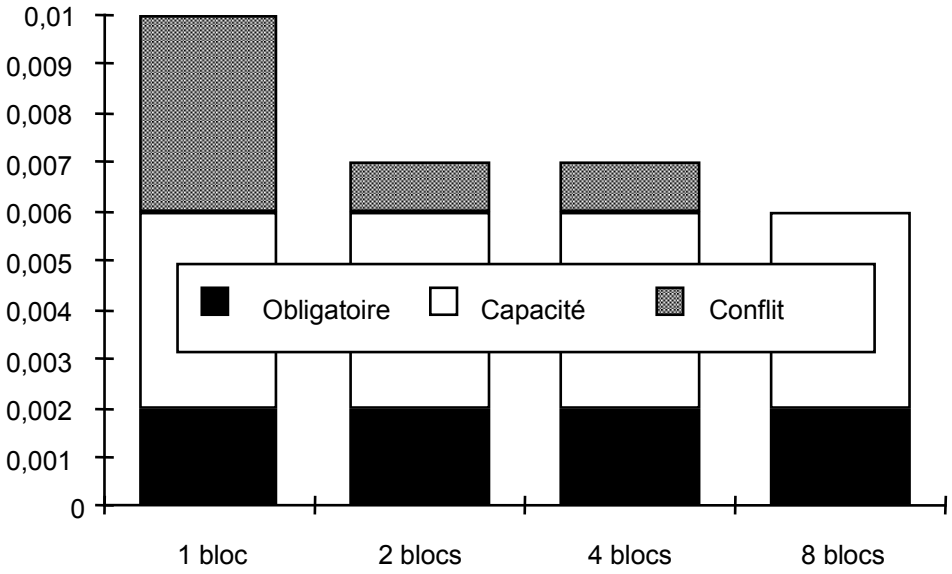
- Echecs de démarrage
- Echecs de capacité
- Echecs de conflit



SPEC92

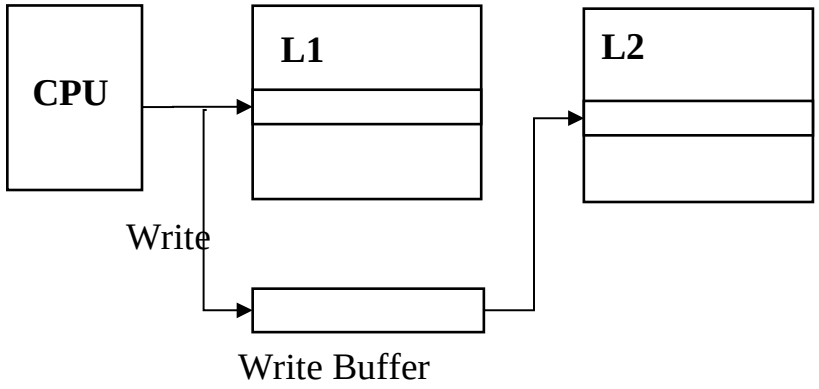
Lignes de 32 octets - LRU

DECStation5000

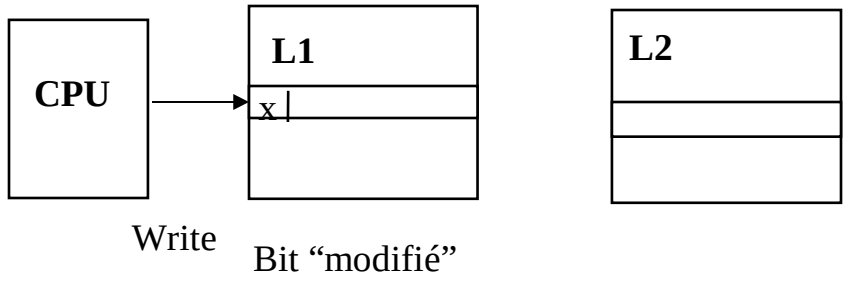


Politique en écriture

WRITE THROUGH
(Ecriture simultanée)



WRITE BACK
(Réécriture)

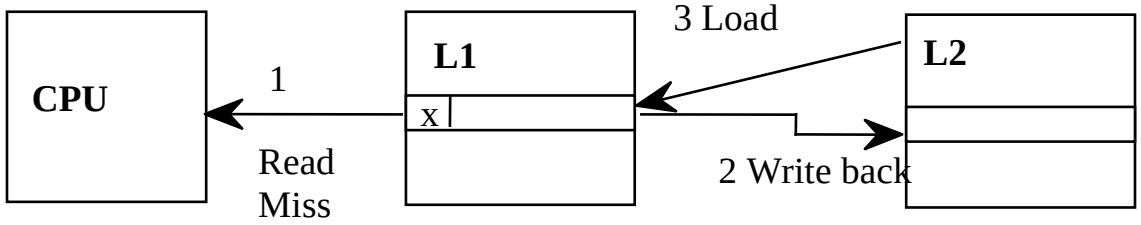


Cohérence
au plus tôt

Cohérence au plus tard

Réécriture

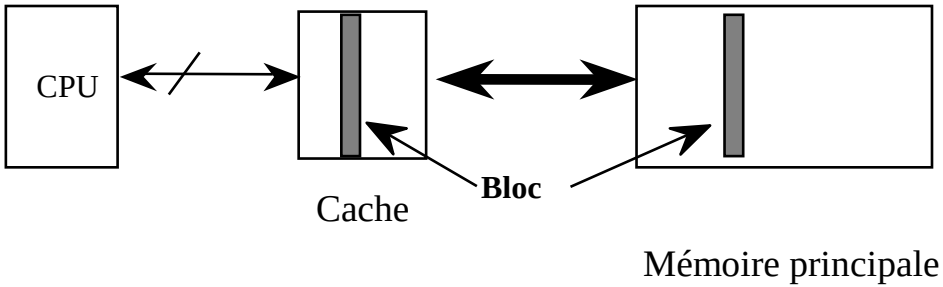
Ecriture mémoire quand une ligne modifiée est remplacée



Politiques de remplacement

- Ligne à remplacer ?
  - La plus anciennement utilisée (LRU)
  - Au hasard
  - La moins fréquemment utilisée (LFU)

CPI Mémoire



$CPI_{mémoire}$ = ma x m x p

ma: accès mémoire/instruction

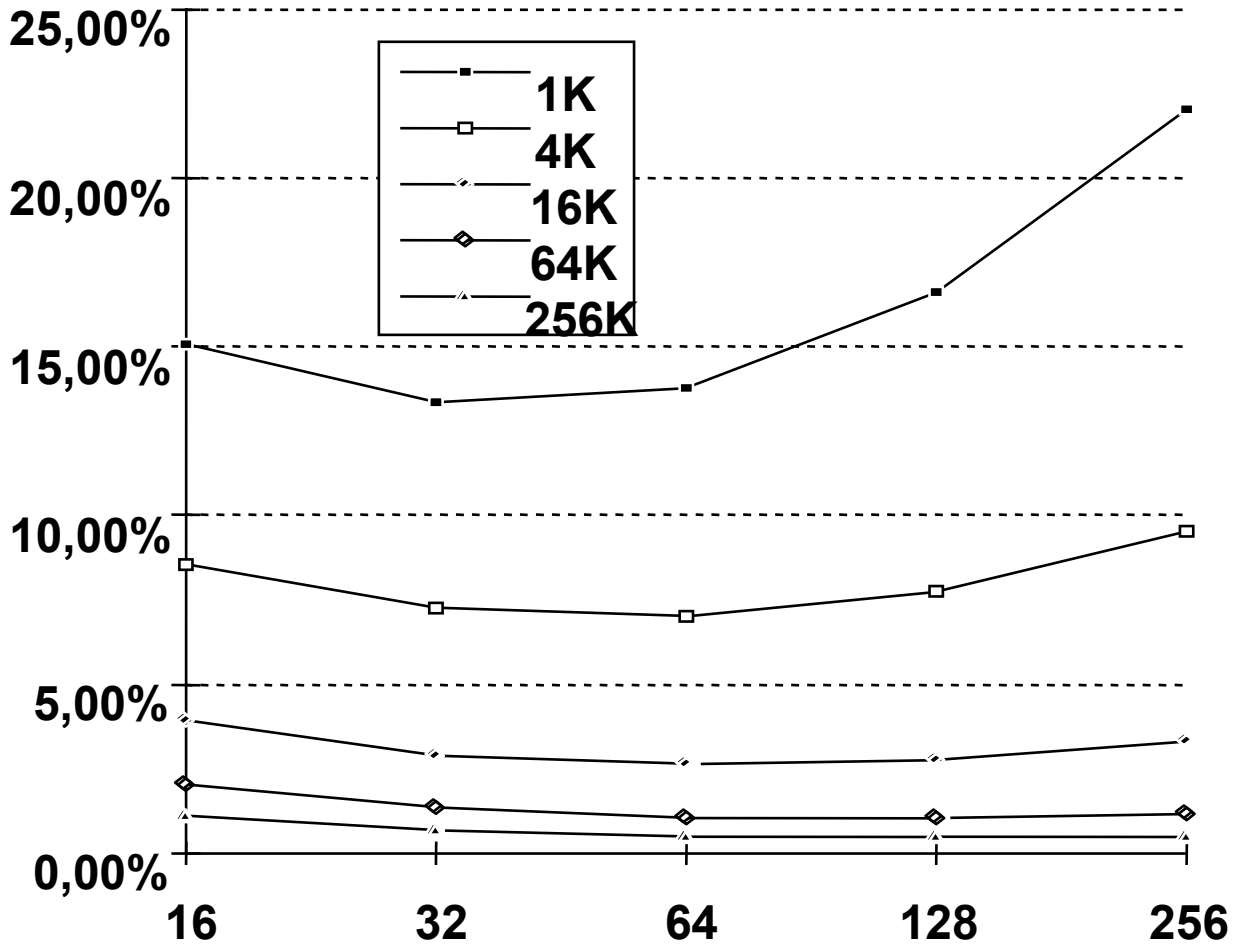
m: taux d'échec

p : pénalité d'échec (cycles d'horloge)

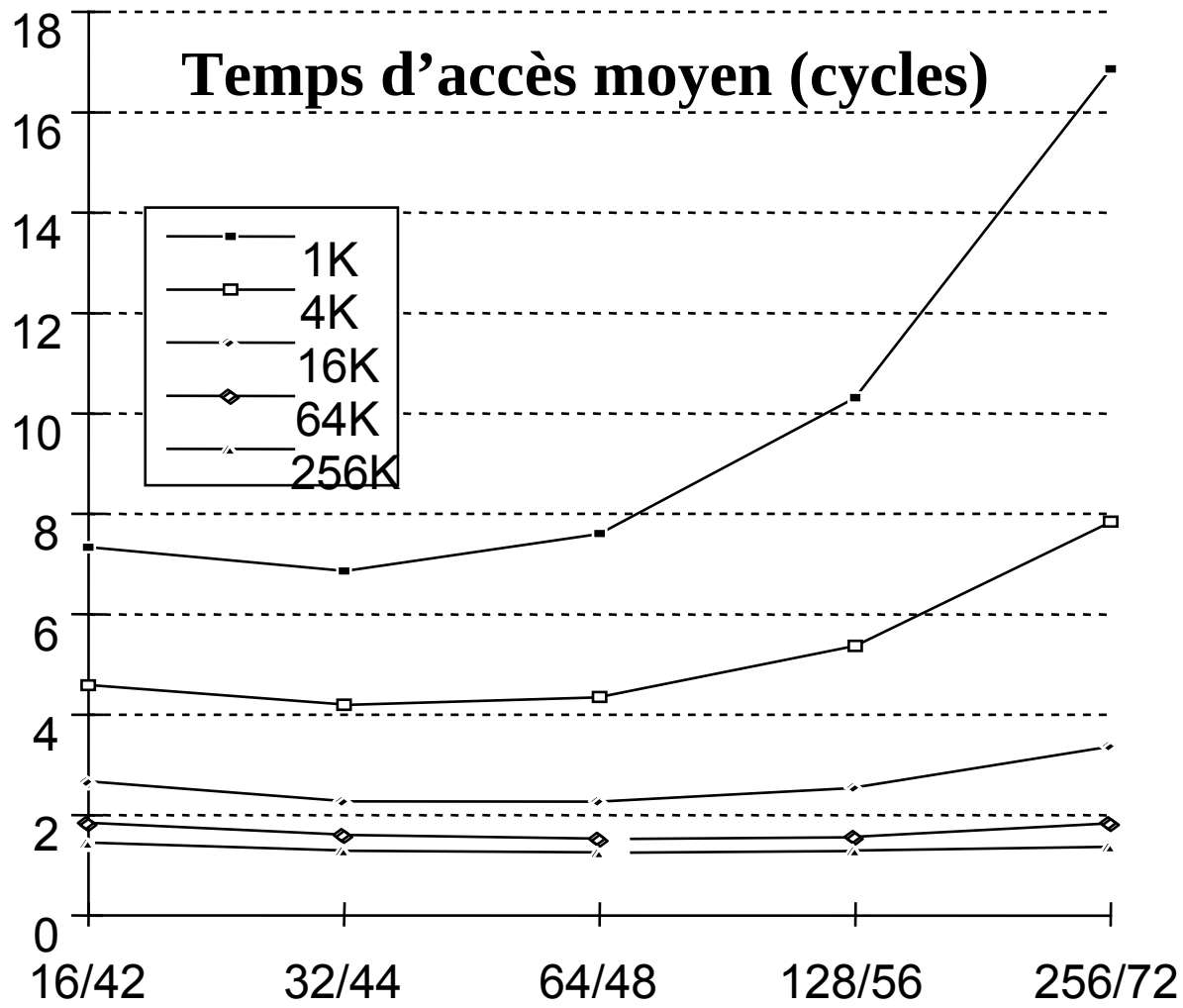
Améliorer les performances des caches

- Diminuer le taux d'échec
  - Taille (cache, ligne) et associativité
  - Optimisations du compilateur
  - Préchargement
- Diminuer le temps d'accès réussi (hit)
  - Petits caches
  - Eviter les traductions d'adresse
- Diminuer la pénalité d'échec
  - Caches non bloquants
  - Caches de second niveau

Taux d'échec (Taille de ligne)



Temps d'accès (taille de ligne)



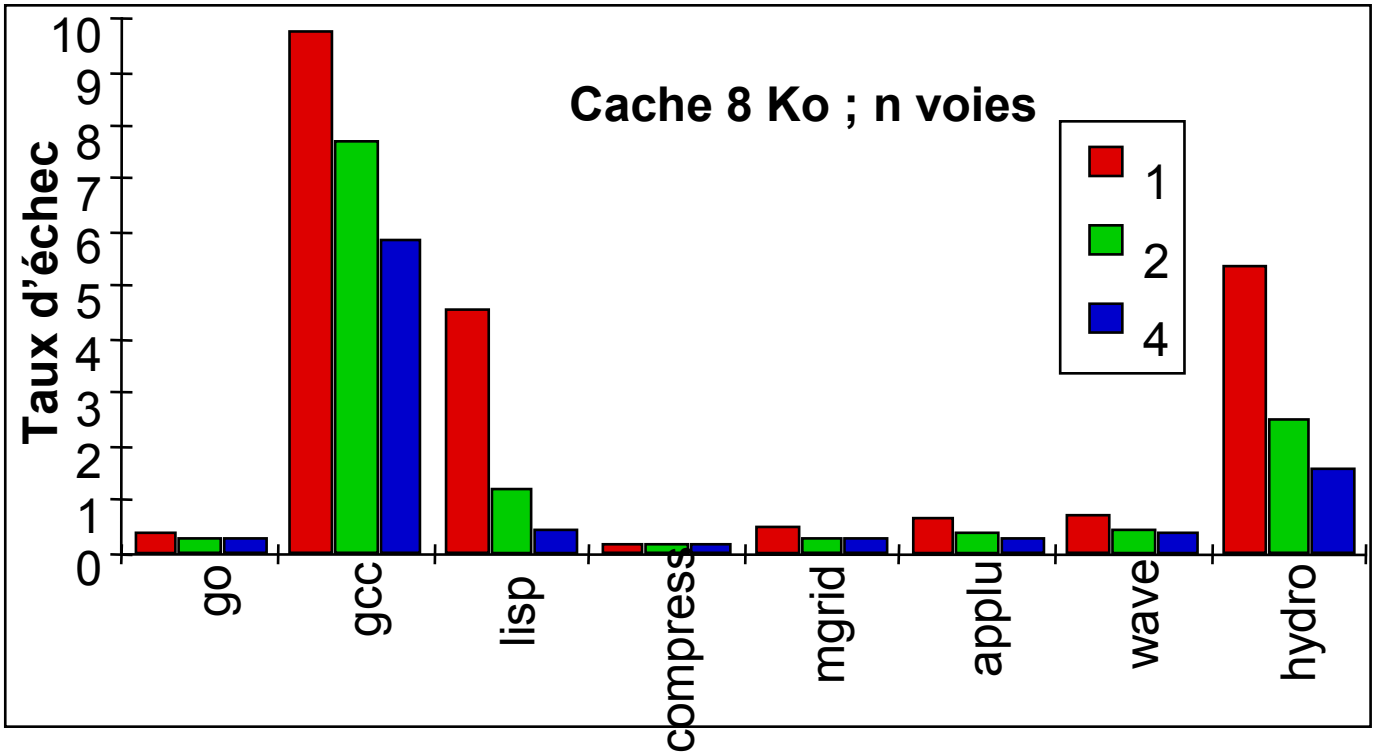
Hypothèse

40 cycles de démarrage, puis 16 octets tous les 2 cycles

Taille de ligne / Pénalité d'échec

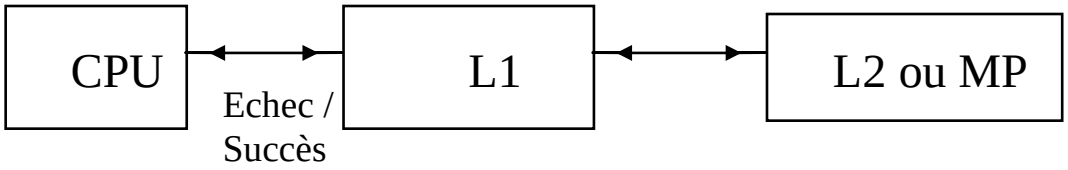
Taux d'échecs (associativité)

Le taux d'échec décroît quand l'associativité croit, mais les lectures "optimistes" ne peuvent plus être utilisées (nécessité de prédicteurs d'ensemble)

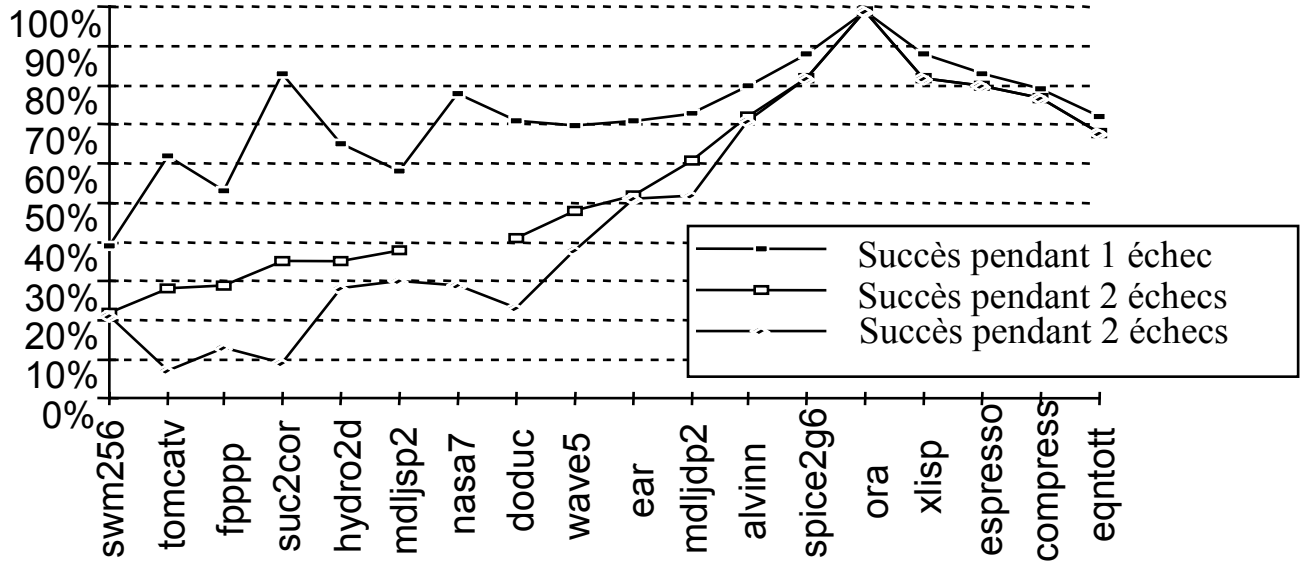


Caches non bloquants

Succès pendant échecs : le cache continue à fournir les données des accès réussis pendant qu'il traite un échec.

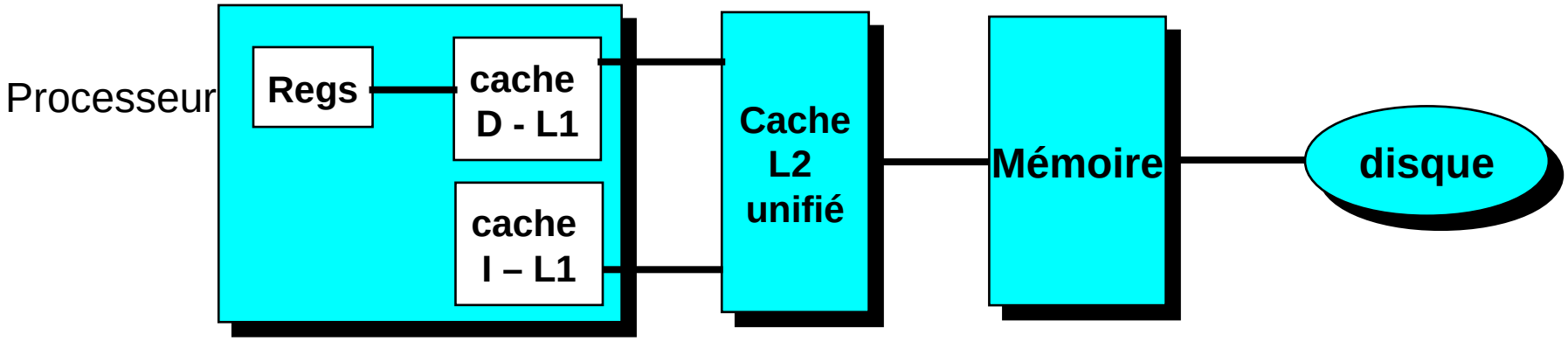


Rapport moyen entre les temps de suspension d'un cache non bloquant/cache bloquant



Plusieurs niveaux de cache

- Options: caches instructions et données séparés ou cache unifié

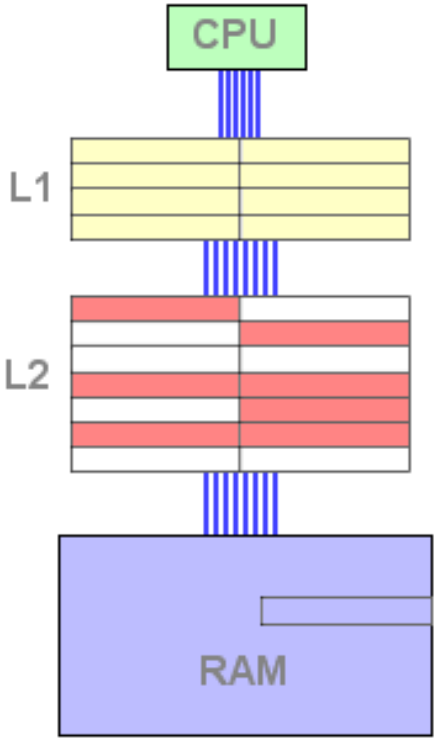


Taille :	200 o	8-64 Ko	SRAM 1-4Mo	DRAM 128 Mo	30 Go
Vitesse	<1 ns	<1 ns	6 ns	60 ns	8 ms
€/Moctet :			100 €/Mo	1.50€/Mo	0.05€/Mo
Taille ligne :	8 o	32 o	32 o	8 Ko	

Plus gros, plus lent, moins cher

Caches inclusifs ou exclusifs

- Caches inclusifs
  - L1$\subset$ L2 $\subset$ MP
- Caches exclusifs
  - L1 $\not\subset$ L2
  - Fonctionnement
    - Echec L1 : bloc de MP dans L1
      - Si remplacement, bloc remplacé dans L2 et bloc demandé dans L1
    - Echec L1 et Succès L2
      - Remplacement d'un bloc dans L1 (copié dans L2) et bloc demandé de L2 dans L1
    - Nécessité d'un tampon des victimes
      - Lors d'un remplacement, le bloc remplacé est placé dans un tampon pour ne pas retarder le transfert L2 vers L1



Caches données des processeurs Intel

CACHE PRIMAIRE

	Taille	Assoc.	Lignes	Latence	Ecriture
Pentium Pro	8 Ko	2 voies	32 oct	3	Réécriture
Pentium III	16 Ko	4 voies	32 oct	3	Réécriture
Pentium 4	8 Ko	4 voies	64 oct	2/6	Simultanée

CACHE SECONDAIRE (UNIFIE)

	Taille	Assoc.	Lignes	Latence	Ecriture
Pentium Pro	512 Ko	4 voies	32 oct		Réécriture
Pentium III*	256 Ko	8 voies	32 oct	6	Réécriture
Pentium 4	256 Ko	8 voies	128 oct	7/7	Réécriture

Amélioration des performances cache

- Réduire le taux d'échec
  - Taille de cache, de ligne et degré d'associativité
  - Optimisations du compilateur — ***HW***
  - Préchargement — ***SW***
- Réduire le temps de l'accès réussi — ***HW*** ***SW***
  - Prédiction d'ensemble
  - Eviter les traductions d'adresse — ***HW***
- Réduire la pénalité d'échec — ***HW***
  - Caches non bloquants
  - Caches de second niveau — ***HW***
    HW

L'allocation mémoire des tableaux C

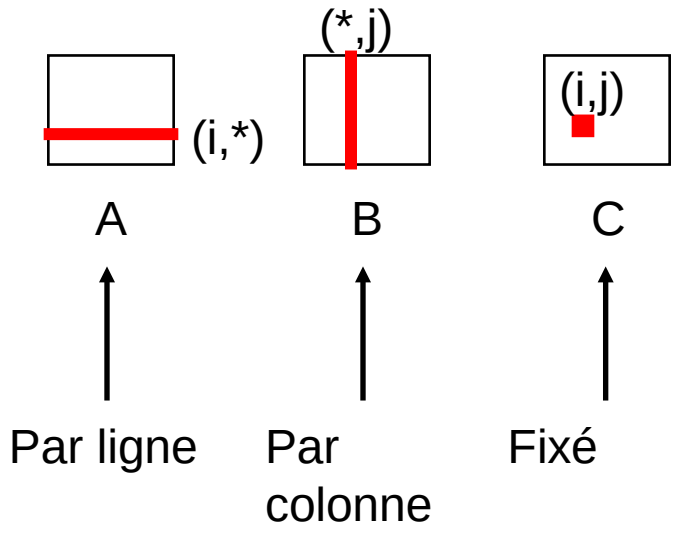
- Les tableaux C sont alloués dans l'ordre ligne d'abord
  - Ligne dans des cases mémoires contiguës
- Balayage à travers les colonnes dans une ligne :
  - ```
    for (i = 0; i < N; i++)
      sum += a[0][i];
    ```
  - Accède aux éléments successifs
  - Si la taille de bloc (B) > 4 octets, exploite la localité spatiale
    - Taux d'échecs obligatoires = 4 octets / B
- Balayage à travers les lignes dans une colonne :
  - ```
    for (i = 0; i < n; i++)
      sum += a[i][0];
    ```
  - Accèdent à des éléments distants
  - Aucune localité spatiale !
    - Taux d'échecs obligatoires = 1 (i.e. 100%)

Multiplication de matrices (ijk)

```
/* ijk */
for (i=0; i<n; i++)  {
  for (j=0; j<n; j++) {
    sum = 0.0;
    for (k=0; k<n; k++)
      sum += a[i][k] * b[k][j];
    c[i][j] = sum;
  }
}
```

PRODUIT SCALAIRE

Boucle interne :



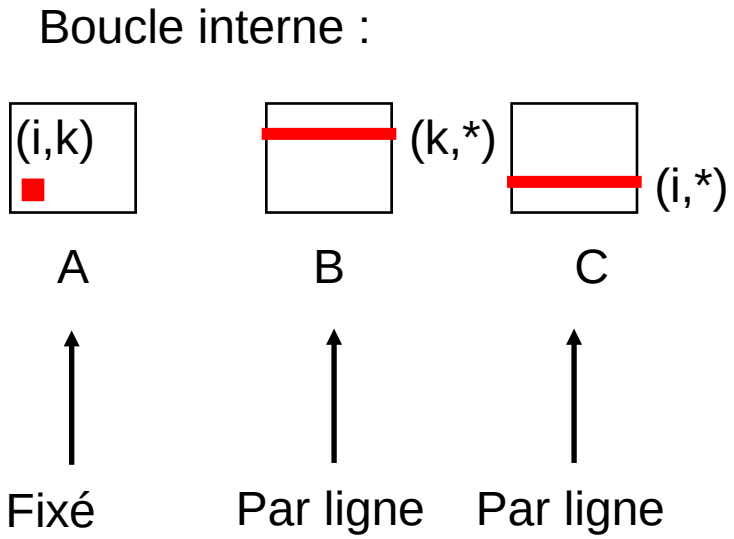
- Echecs par itération de la boucle interne:

A	B	C
0,25	1,0	0,0

Multiplication de matrices (ikj)

```
/* ikj */
for (i=0; i<n; i++) {
  for (k=0; k<n; k++) {
    r = a[i][k];
    for (j=0; j<n; j++)
      c[i][j] += r * b[k][j];
  }
}
```

SAXPY/DAXPY



- Echecs par itération de la boucle interne:

A	B	C
0,0	0,25	0,25

Améliorer la localité temporelle par blocage

- Exemple : multiplication de matrices avec blocage
  - "bloc" (dans ce contexte) ne signifie pas "bloc de cache".
  - C'est un sous bloc de la matrice.
  - Exemple: N = 8; taille de sous bloc = 4

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Idée de base : Un sous bloc (i.e., $\mathbf{A_{xy}}$) peut être traité comme un scalaire.

$$C_{11} = A_{11}B_{11} + A_{12}B_{21} \qquad C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

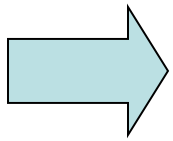
$$C_{21} = A_{21}B_{11} + A_{22}B_{21} \qquad C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Optimisations du compilateur

ECHANGE DE BOUCLES

Améliore la localité spatiale

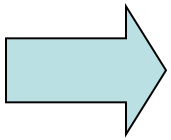
```
for (j=0; j<100; j++)
    for (i=0; i<5000; i++)
        x[i][j] = 2*x[i][j];
```



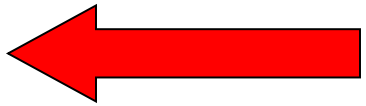
```
for (i=0; i<5000; i++)
    for (j=0; j<100; j++)
        x[i][j] = 2*x[i][j];
```

FUSION DE TABLEAUX

```
int val[SIZE];
int key[SIZE];
```



```
struc merge {
        int val;
;       int key; }
struc merge merge-array [SIZE];
```



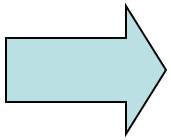
Optimisations du compilateur

Améliore la localité temporelle

FUSION DE BOUCLES

```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        a[i][j] = 1/b[i][j]*c[i][j];

for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        d[i][j] = a[i][j]+c[i][j];
```



```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
      {
        a[i][j] = 1/b[i][j]*c[i][j];
        d[i][j] = a[i][j]+c[i][j];
      }
```

Optimisations du compilateur

BLOCAGE

AMELIORE LA LOCALITE TEMPORELLE

```
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        {r=0;
        for (k=0; k<N; k++)
            r+=y[i][k]*z[k][j];
        x[i][j]=r;}
```

```
for (jj=0; jj<N; jj+=B)
for (kk=0; kk<N; kk+=B)
for (i=0; i<N; i++)
    for (j=jj; j<(min(jj+B-1,N); j++)
        {r=0;
        for (k=kk; j<(min(kk+B-1,N); k+
+)
            r+=y[i][k]*z[k][j];
        x[i][j]+=r;}
```

B : facteur de blocage

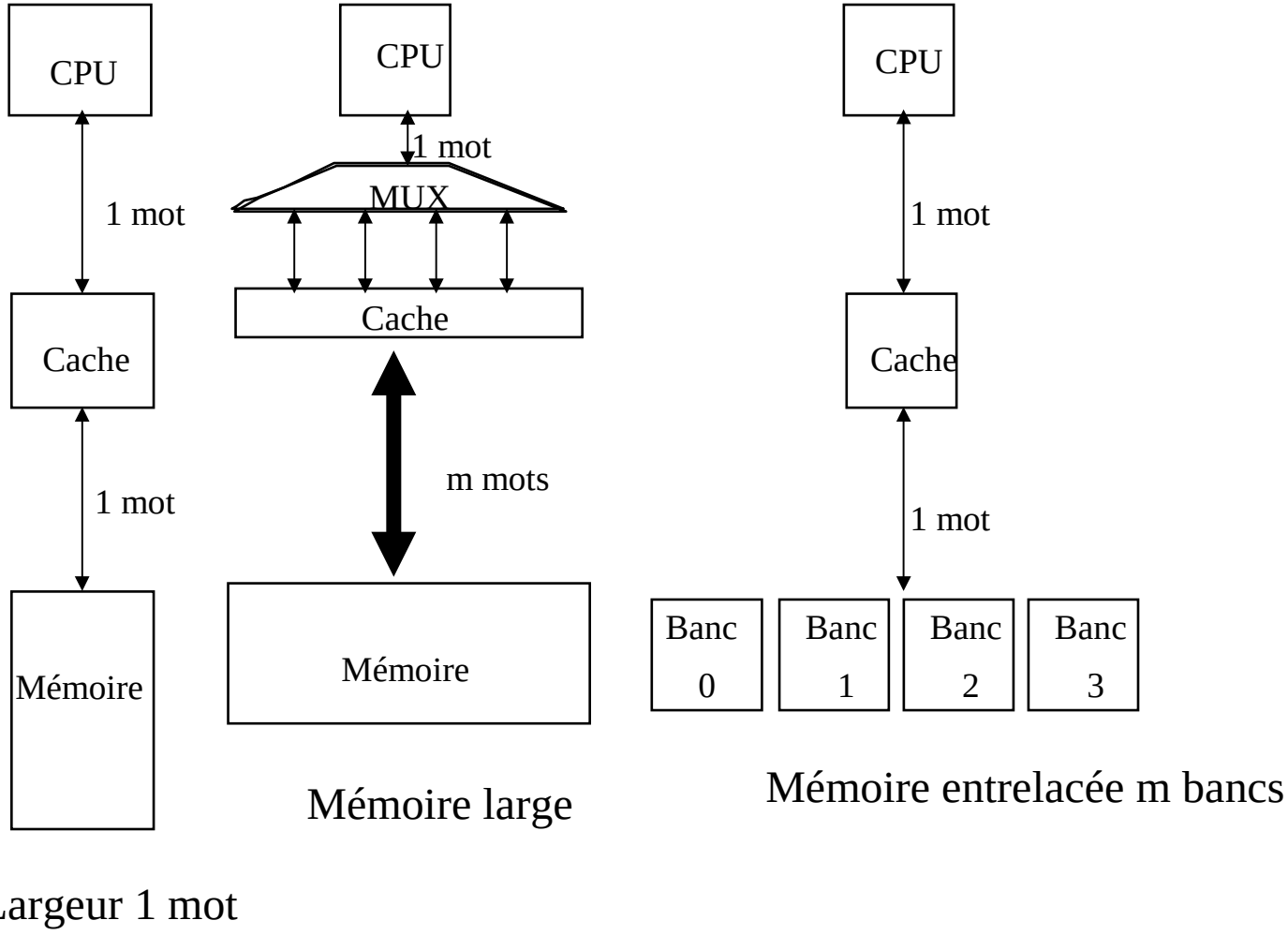
Préchargement matériel

- Précharger les instructions ou les données avant qu'elles ne soient nécessaires (charger une ligne avant un échec cache)
- Quelle ligne ?
  - La suivante (préchargement séquentiel selon la localité spatiale)
  - Une ligne prédite
- Quand ?
  - Toujours
  - Sur un échec
  - Sur un échec et lorsqu'on accède une donnée préchargée
- Où ?
  - Dans le cache (pollution potentielle)
  - Dans un tampon

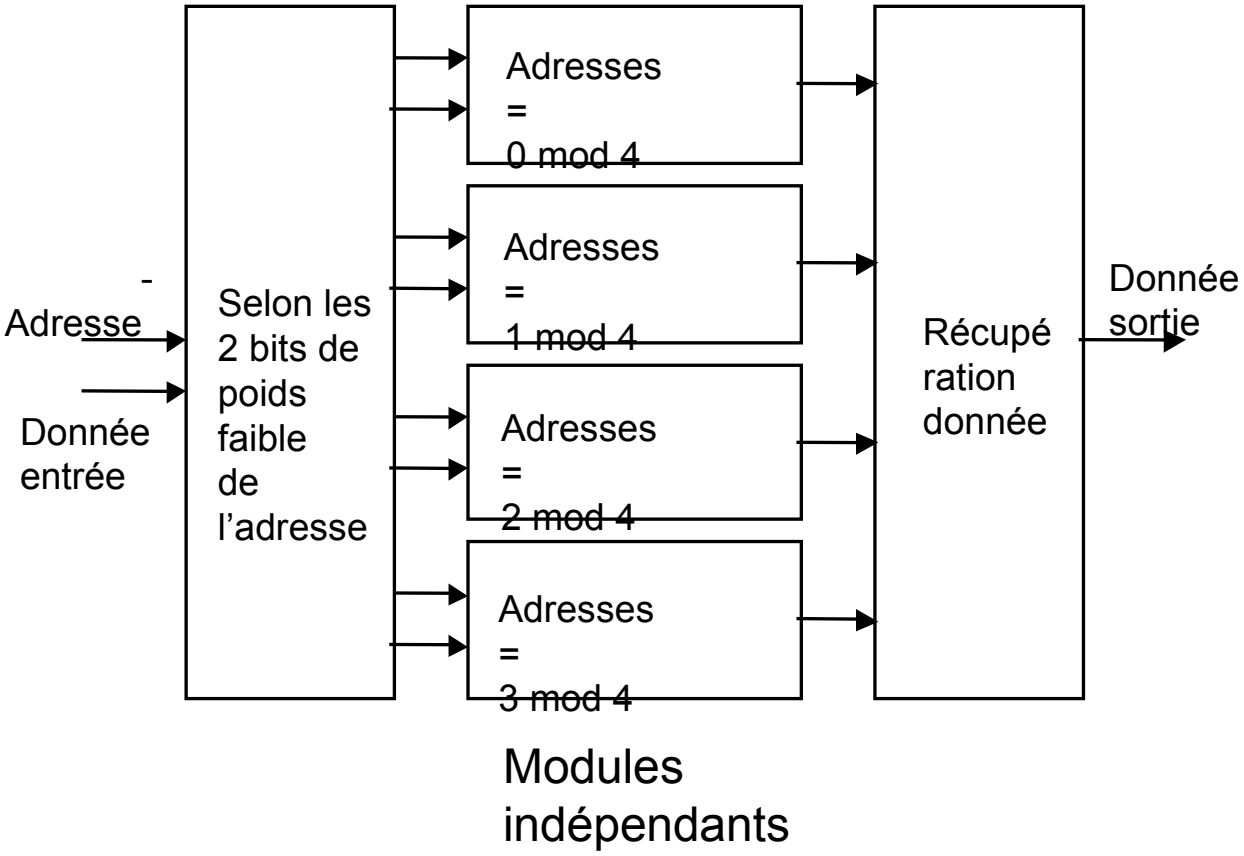
Préchargement logiciel

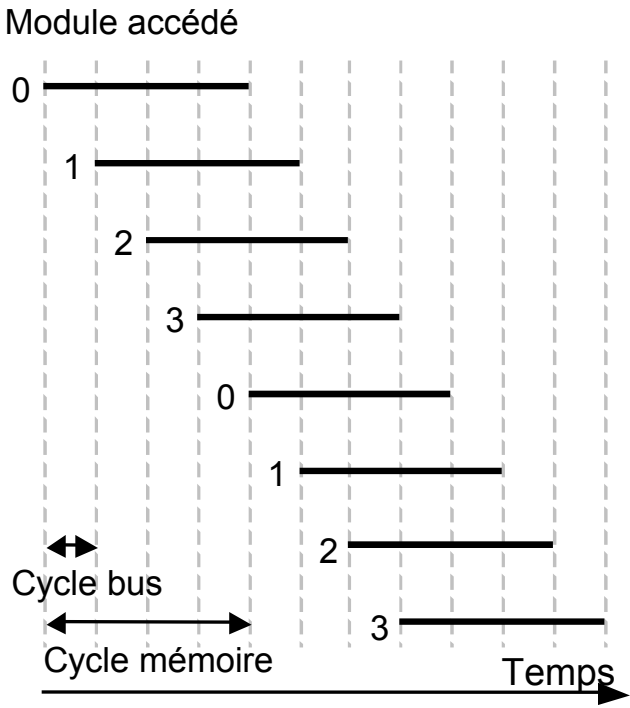
- Instructions de préchargement (IA-32)
  - PREFETCHT0 m8 : tous les niveaux de cache
  - PREFETCHT1 m8: tous les niveaux sauf L0
  - PREFETCHT2 m8: tous les niveaux sauf L0 et L1
  - PREFETCHNTA m8: préchargement dans une structure non temporelle
- Indications de préchargement
  - Dépendent des implémentations

Liaison cache-mémoire principale



Entrelacement mémoire

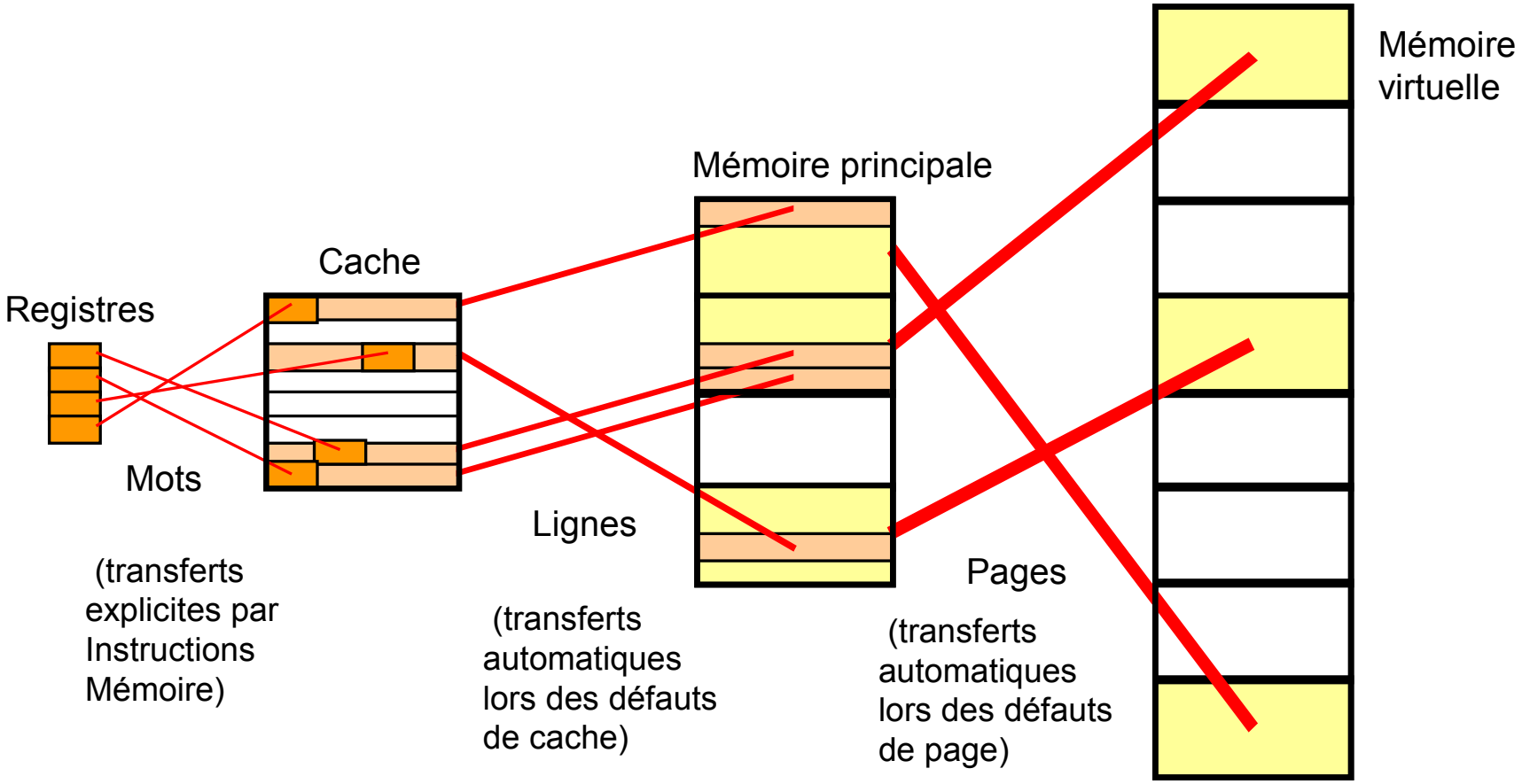




Mémoire virtuelle

- Beaucoup de programmes se partagent la mémoire DRAM
- On peut écrire des programmes sans tenir compte de la taille de la mémoire principale
- Relocation : des parties de programme peuvent être placés à différents endroits mémoire
- Mémoire virtuelle
  - La mémoire DRAM contient plusieurs programmes s'exécutant en même temps (processus)
  - On utilise la mémoire DRAM comme une sorte de "cache" pour le disque

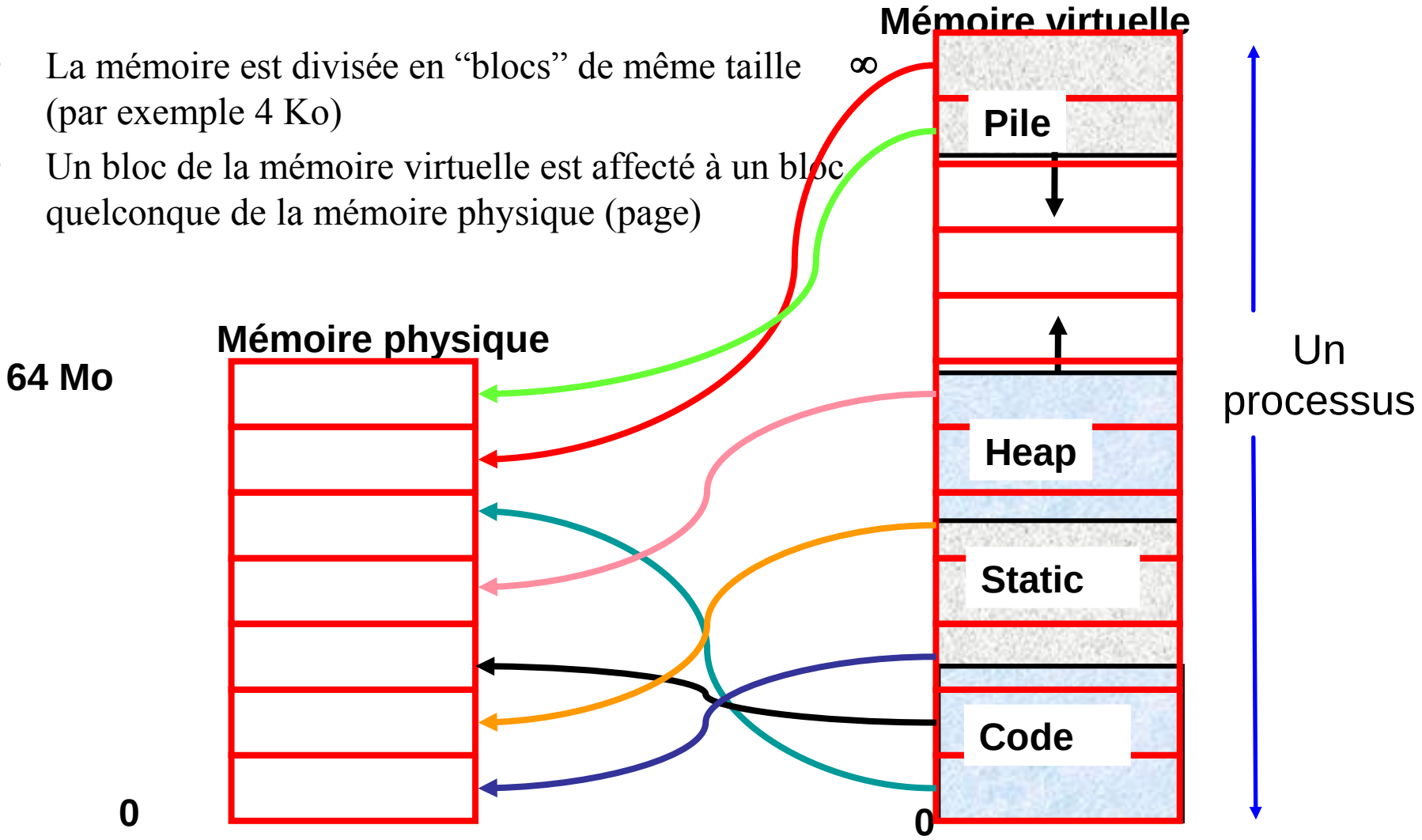
Plan d'ensemble de la hiérarchie mémoire



Transferts de données dans une hiérarchie mémoire

Mappage mémoire virtuelle sur mémoire physique

- La mémoire est divisée en “blocs” de même taille (par exemple 4 Ko)
- Un bloc de la mémoire virtuelle est affecté à un bloc quelconque de la mémoire physique (page)



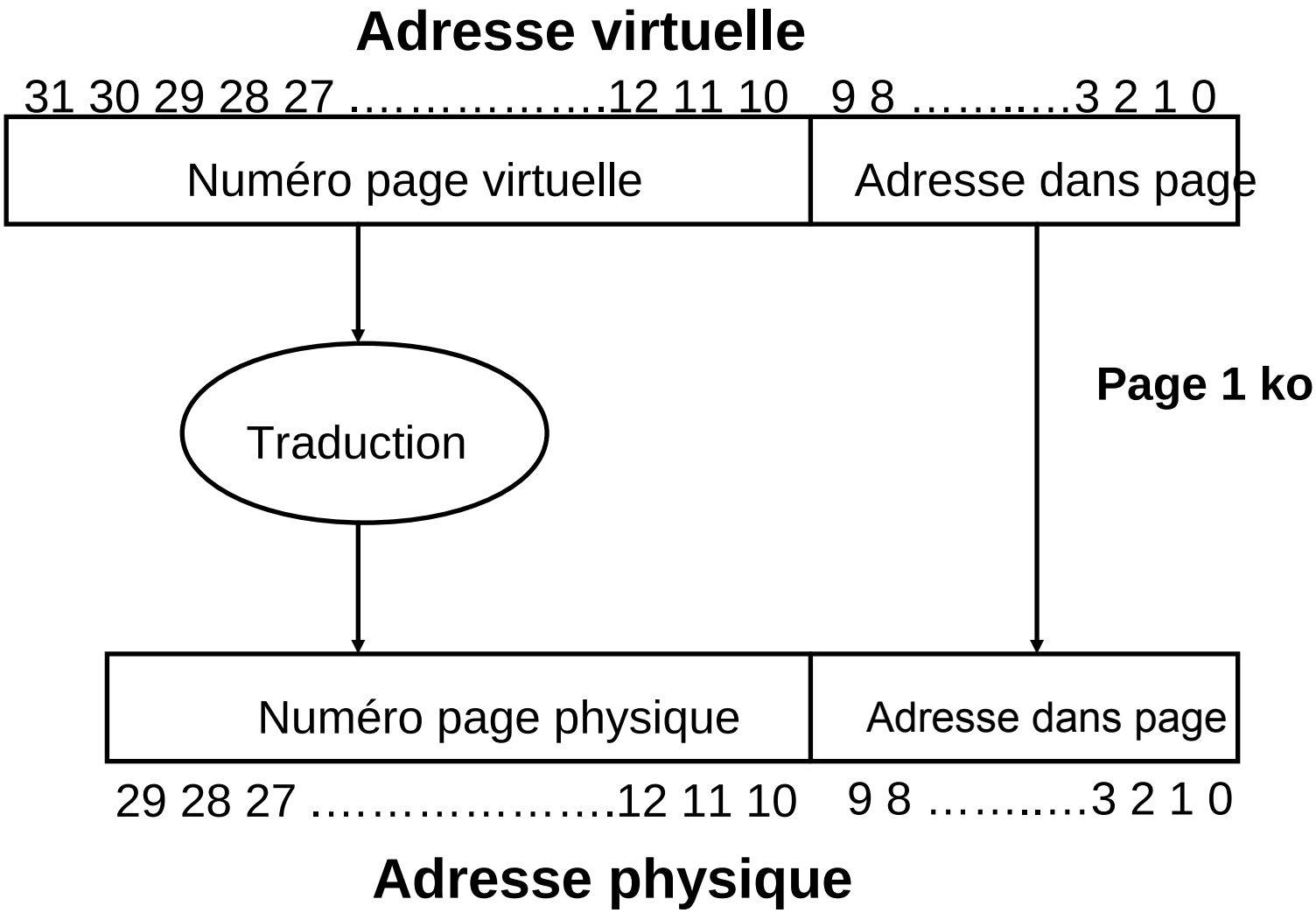
Gestion des défauts de page

- Un défaut de page est comme un échec cache
  - On doit trouver la page dans le niveau inférieur de la hiérarchie
- Si le bit « valide » est à 0, le numéro de page physique pointe vers une page sur disque
- Quand le système d'exploitation démarre un nouveau processus, il crée de l'espace sur disque pour toutes les pages du processus, positionne tous les bits « valide » dans la table des pages à 0 et les numéros de pages physiques pour qu'ils pointent sur le disque
  - Les pages du processus sont chargés depuis le disque uniquement quand elles sont nécessaires (pagination à la demande)

Comparaison Cache - Mémoire virtuelle

	Cache	*Mémoire virtuelle*
Elément transféré	Bloc ou ligne	Page
Echec	Echec cache	Défaut de page
Taille transfert	Bloc : 32-64 octets	Page : 4 ko-16 ko
Placement	Correspondance directe Associativité N voies	Associativité totale
Remplacement	LRU ou hasard	Approximation LRU
Ecriture	Simultanée ou réécriture	Réécriture
Gestion	Matériel	OS (matériel+logiciel

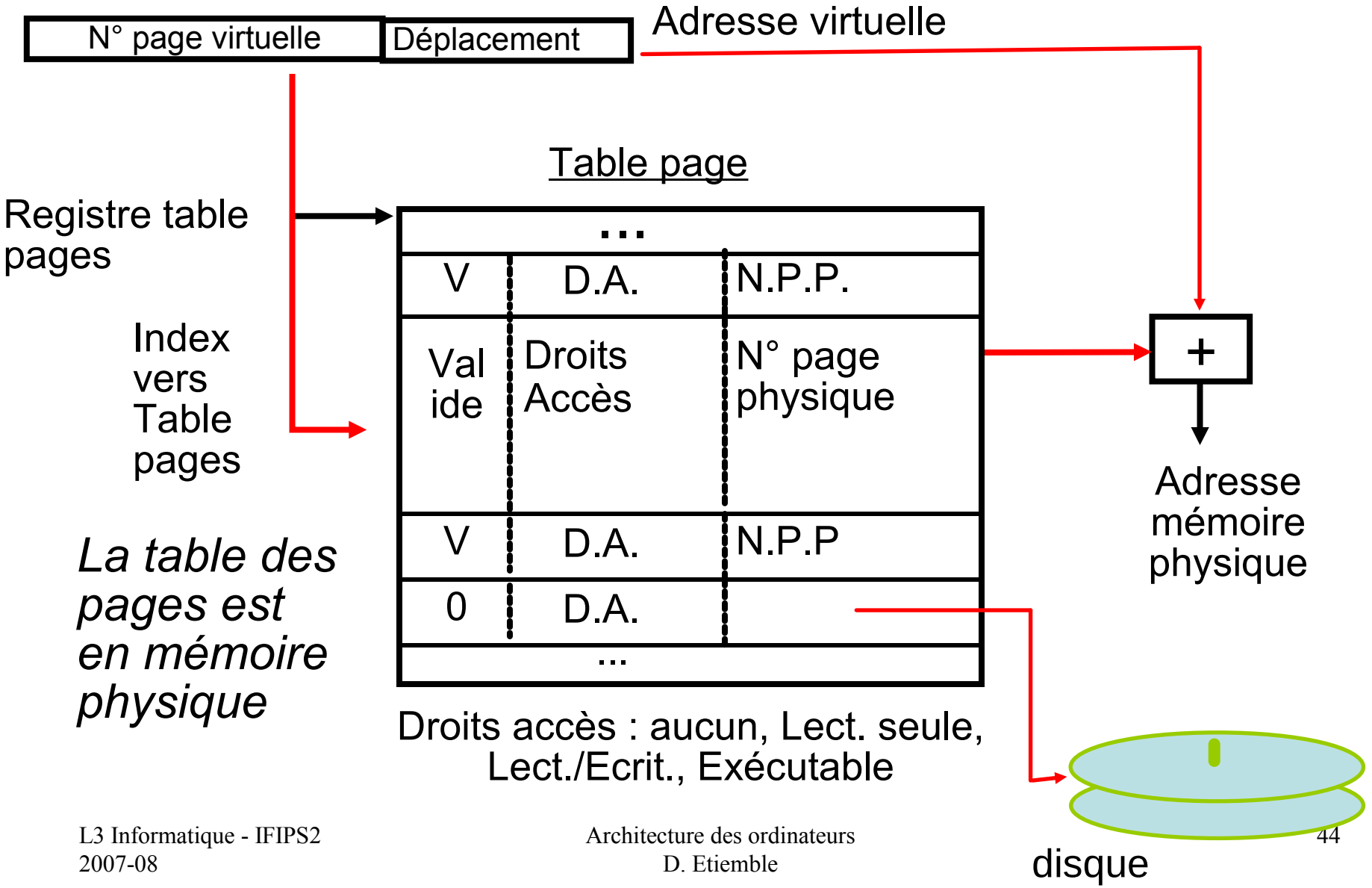
Traduction adresse virtuelle en adresse physique



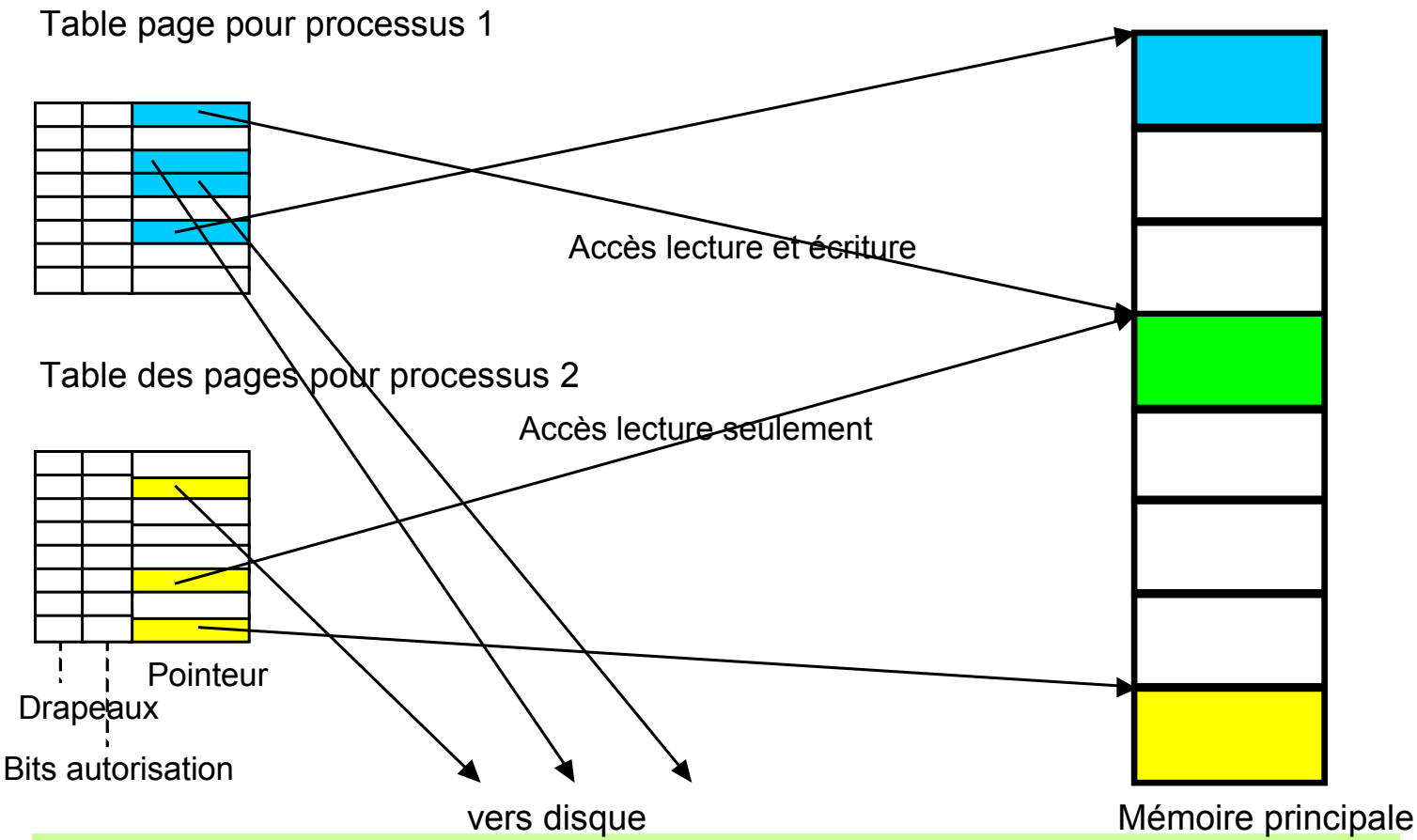
Traduction d'adresse

- Associativité totale pour le placement des pages
- Une table des pages est une structure de données qui contient le mappage des pages virtuelles sur les pages physiques
  - Différentes techniques
- Chaque processus en cours d'exécution a sa propre table des pages

Traduction d'adresse : Table des pages



Protection et partage en mémoire virtuelle



La mémoire virtuelle facilite le partage et la protection mémoire

Table des pages directes à 2 niveaux

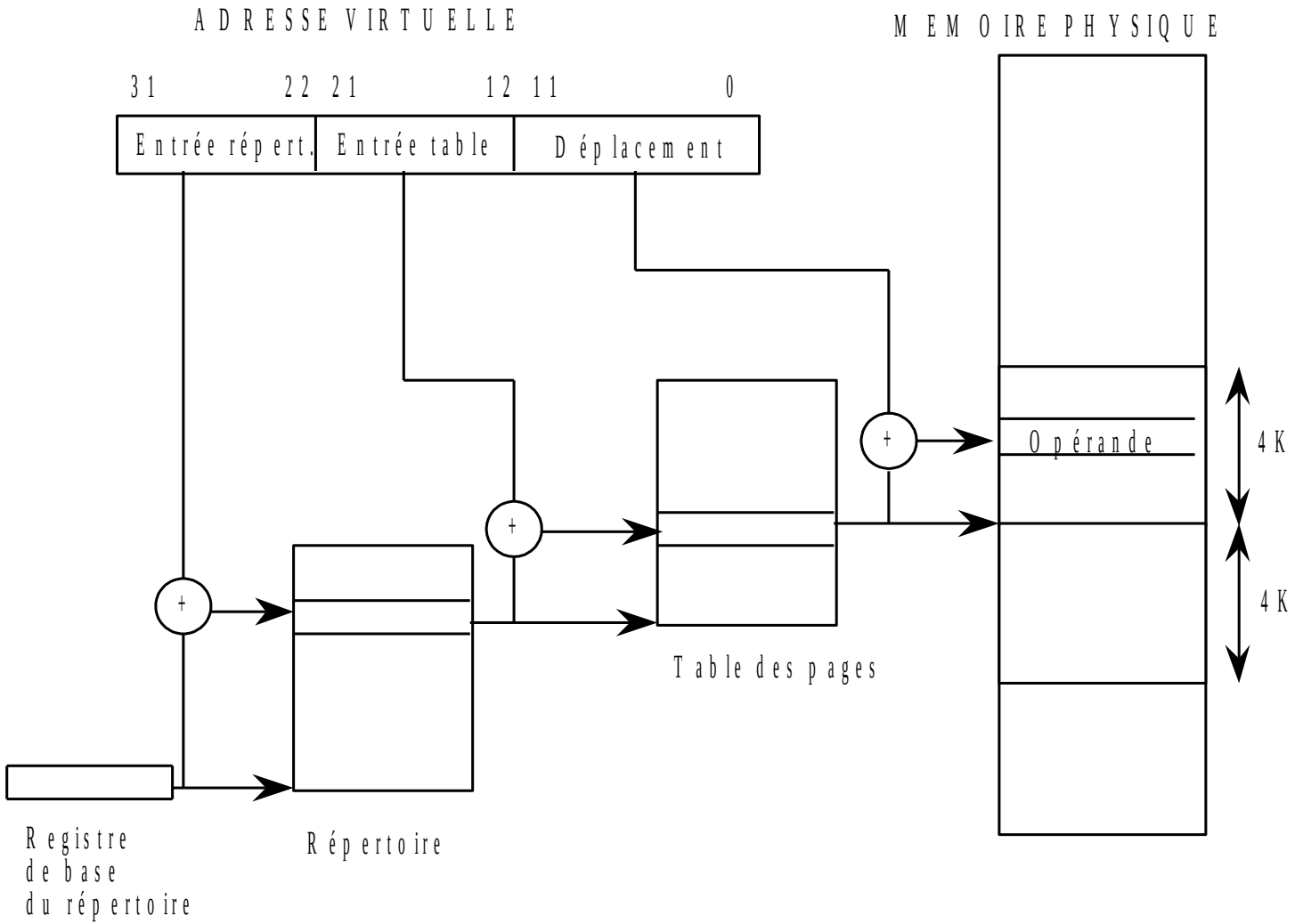
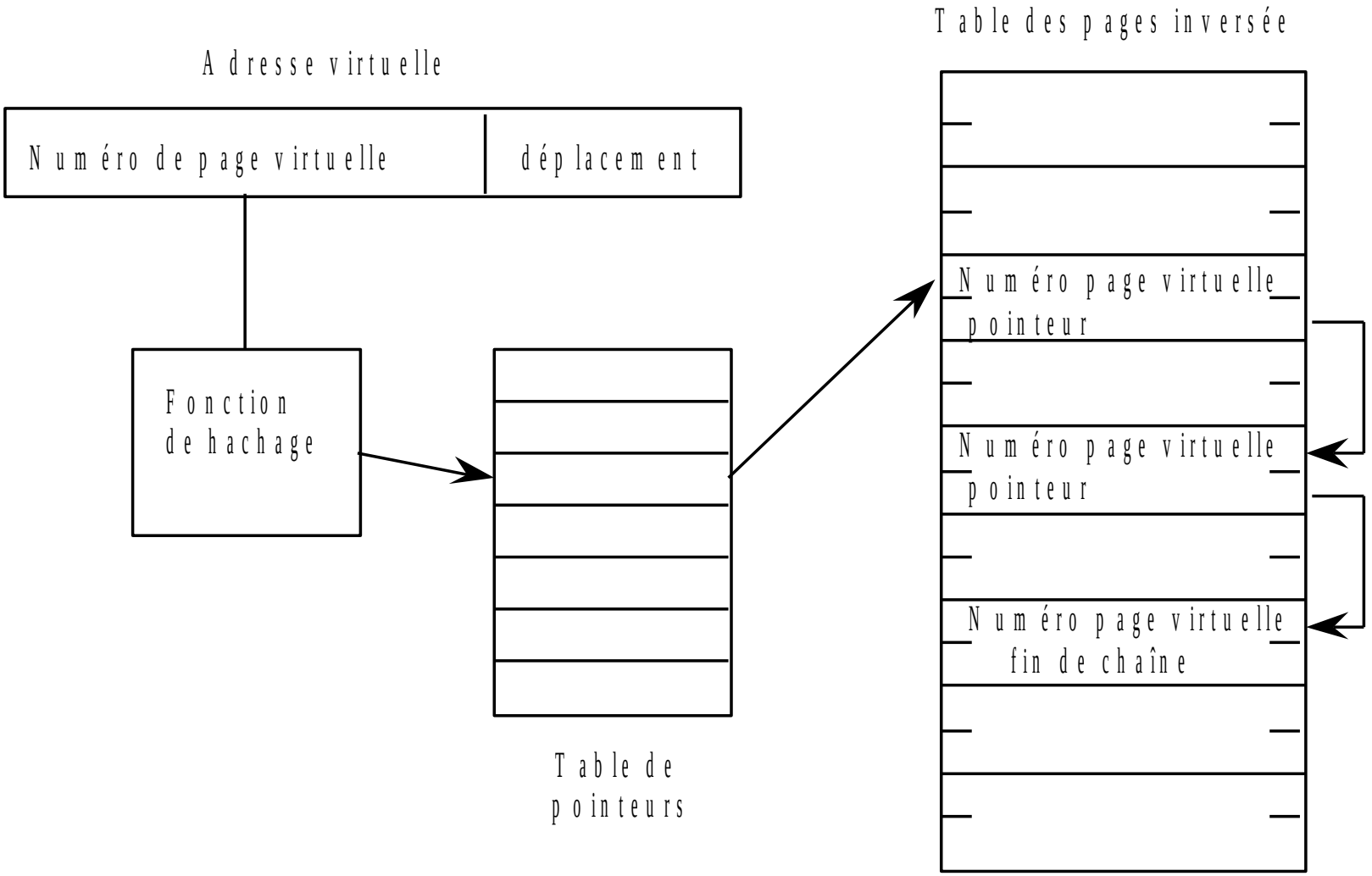


Table des pages inverses



Traduction rapide : TLB

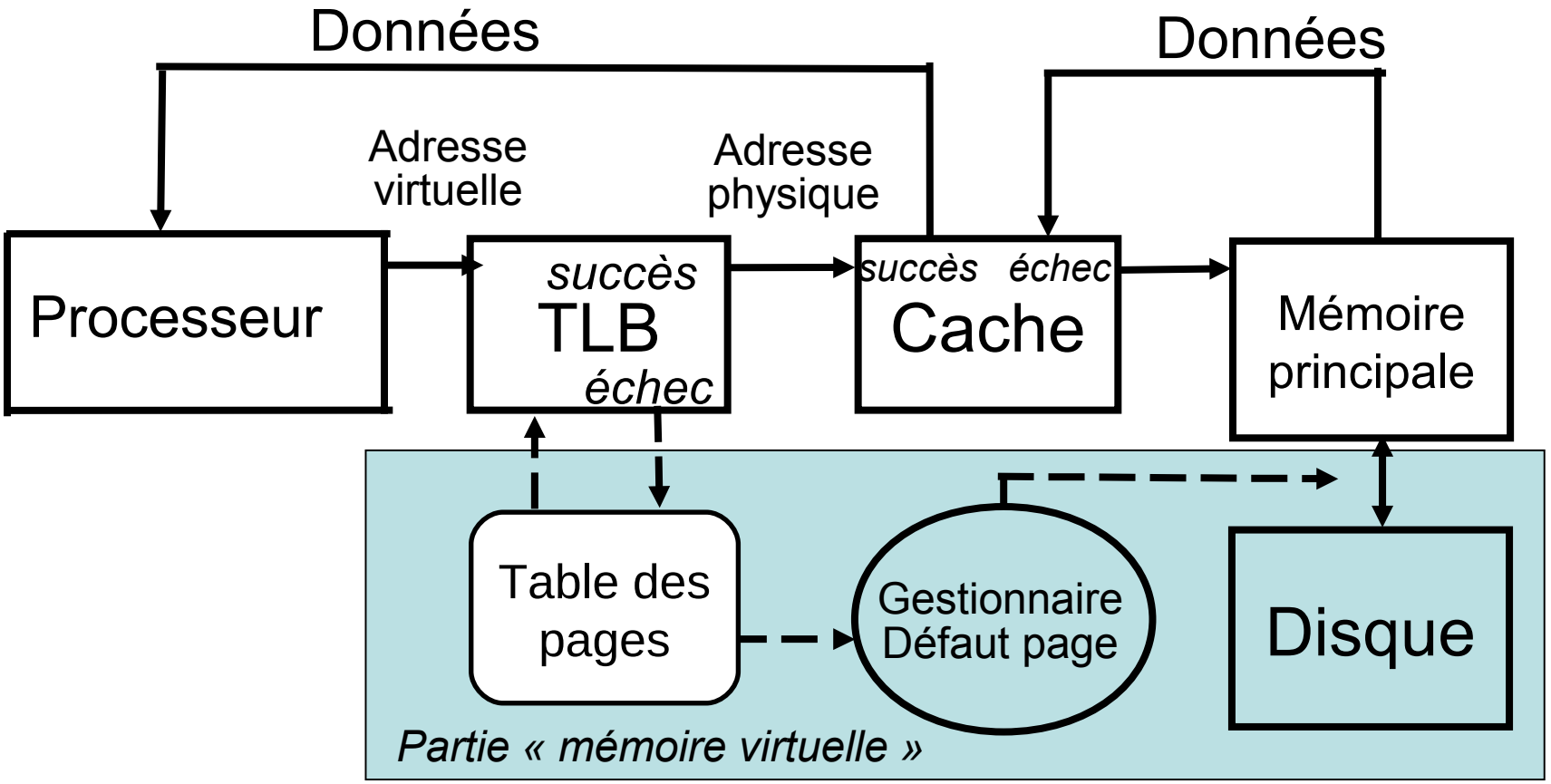
- Problème : la mémoire virtuelle demande 2 (3) accès mémoire
  - 1 (2) pour traduire l'adresse virtuelle en adresse physique (accès à la table des pages)
  - 1 pour l'accès à la donnée réelle (succès cache)
  - La table des pages étant en mémoire physique, il y a donc 2 (3) accès mémoire
- Observation comme il y a de la localité dans les pages de données, il doit y en avoir dans les adresses virtuelles de ces pages
- Créer un cache de traduction des adresses virtuelles en adresses physiques
- Un tel cache de « table des pages » est appelé TLB (Translation Lookaside Buffer)

Format TLB

N° page virtuelle	N° page physique	Valide	Référence	Modifié	Droits d'accès

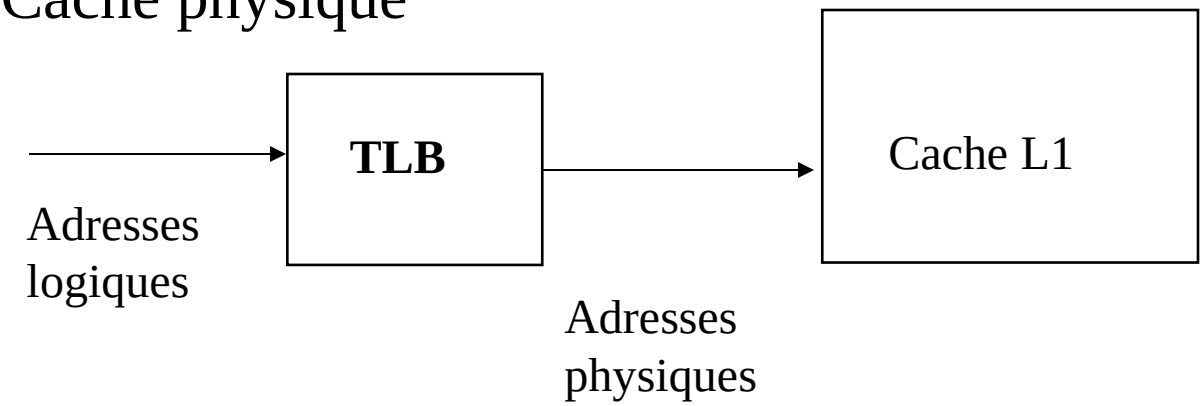
- Modifié : indique si la page a été ou non modifiée en écriture
- Référence : utilisée pour calculer le LRU
- Droits d'accès : lecture/écriture/exécution sur les pages

Fonctionnement du TLB

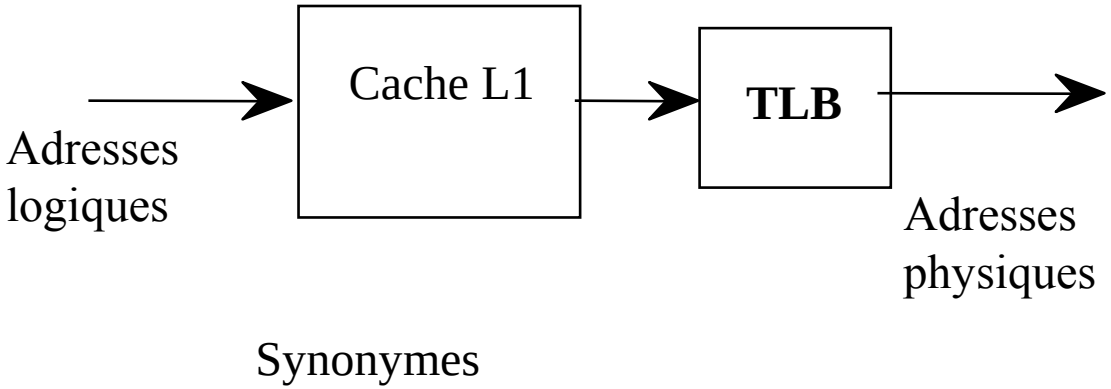


Cache et TLB

Cache physique

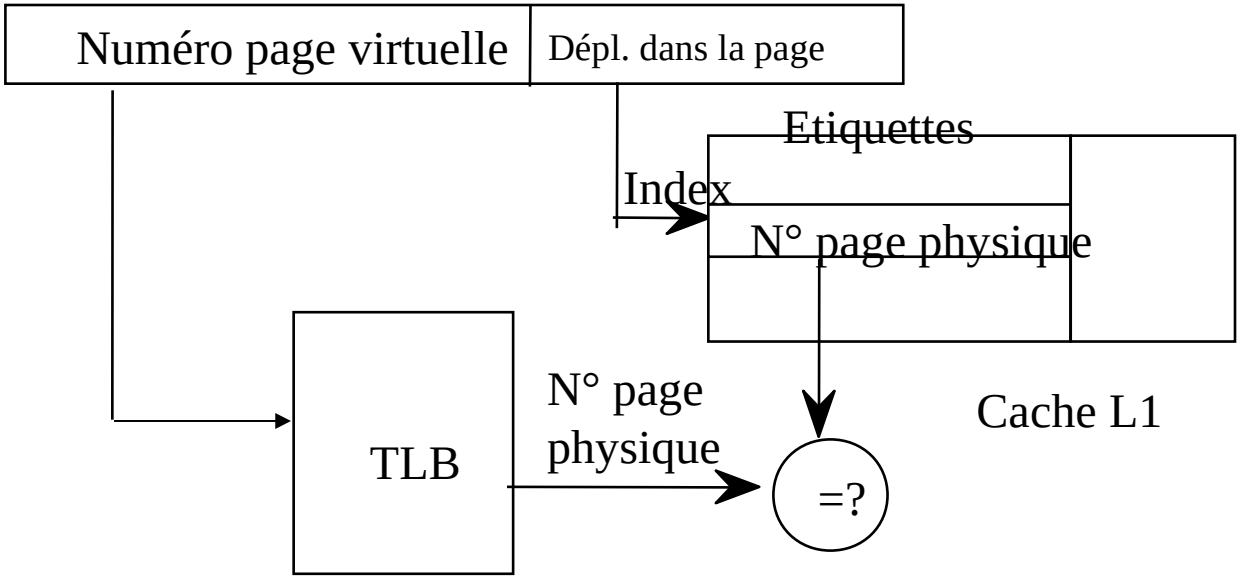


Cache virtuel



Cache et TLB

Caches avec index virtuels et adresses physiques



TLB et cache sont accédés en parallèle

Avec la correspondance directe, taille du cache = taille de page