Master SEM

Concepts avancés d'Architecture

Cours 2.1 : Mémoires caches

Année 2020-2021

Pr. R. BOUDOUR

Mémoire cache

- Exemple : CPU à 2,6GHz et mémoire avec 10ns de latence (temps de calcul de l'adresse avant de transmettre le contenu)
  - Si le CPU veut accéder à une information mémoire, après combien de temps va-t-il recevoir l'information ?

    => 10×10^{-9} s
  - Combien de cycles CPU pendant cette durée ?

    => $10 \times 10^{-9} \times 2{,}6 \times 10^{9} = 26$ cycles

Mémoire cache

- Conclusion:
  - Même si le CPU peut faire n'importe quelle instruction (add, mult…) en quelques cycles, il doit attendre 26 cycles pour avoir les opérandes
  - Trop long pour le CPU d'aller chercher l'information en mémoire ; mais c'est pourtant là qu'elle est !

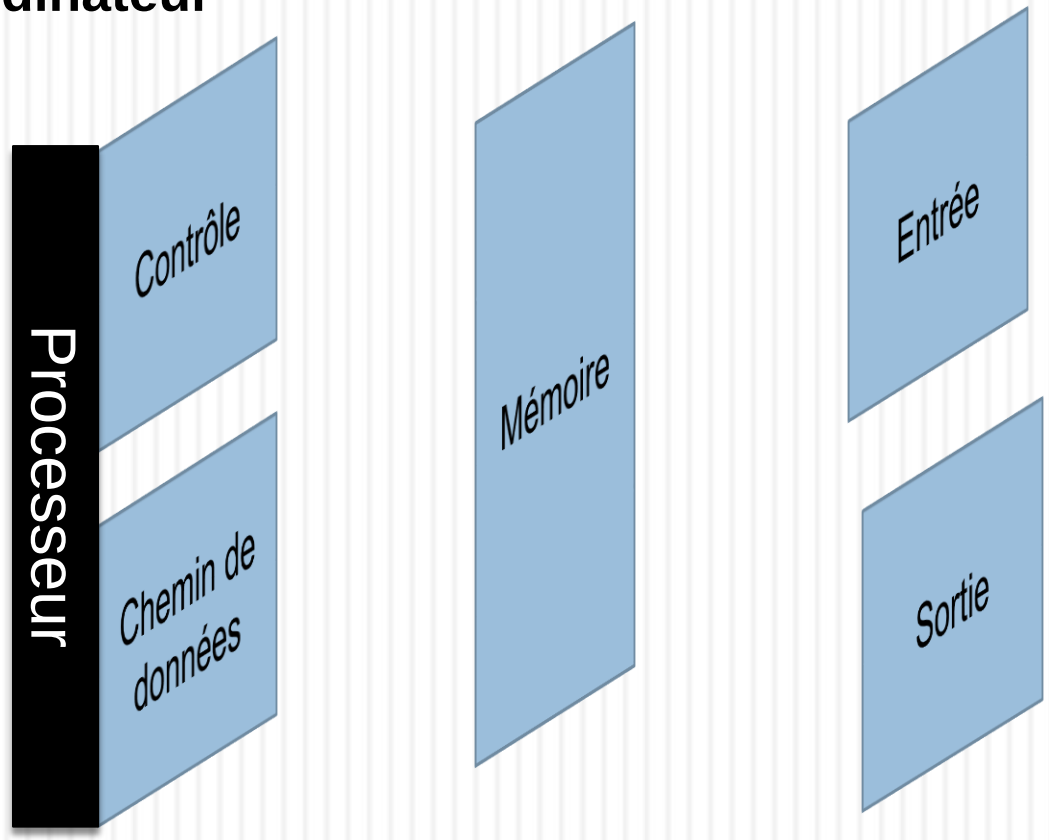
- Solution:
  - Augmenter la capacité de mémoire dite rapide

  La mémoire rapide est chère, donc trop coûteux d'avoir toute la RAM très rapide

=> on va se contenter d'une petite partie du contenu

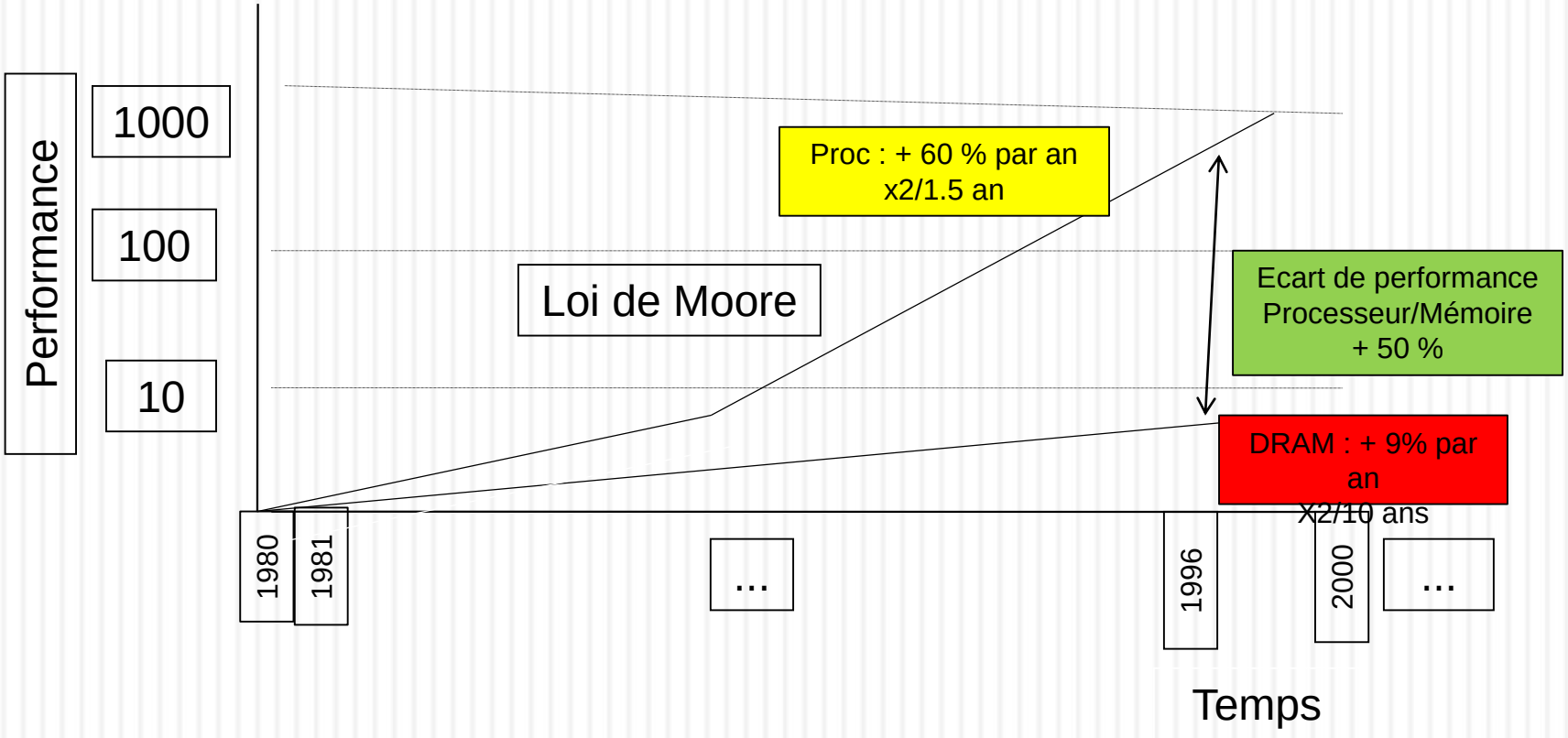
Rappels

- **Un ordinateur**



Ecart entre processeur et mémoire

❑ Mémoire : SRAM et DRAM



Latence mémoire longue

- 60-100 ns not totally uncommon
- Quick back-of-the-envelope calculation :
  - 2 GHz CPU
  - T = 1/f = 0.5 ns / cycle
  - 100 ns memory → 200 cycle memory latency !

Solution : Caches

Principes quantitatifs de conception des ordinateurs

- **Rendre rapide le cas courant**
  - **Loi d'Amdahl**
    - **Accélération = performance pour une tâche entière en utilisant la caractéristique quand c'est possible / performance pour une tâche entière sans utiliser caractéristique**
    - **Accélération** = Temps d'exécution pour la tâche sans utiliser la caractéristique / Temps d'exécution pour la tâche en utilisant la caractéristique quand c'est possible

Principes quantitatifs de conception des ordinateurs

- **Temps exécution** $_{\textbf{nouveau}}$ = temps exécution $_{\text{ancien}}$ x((1 - fraction $_{\text{améliorée}}$) + (fraction $_{\text{améliorée}}$ /accélération $_{\text{améliorée}}$))

- **Accélération** $_{\textbf{globale}}$ = Temps exécution $_{\text{ancien}}$ / Temps exécution $_{\text{nouveau}}$

  = 1/ ((1-Fraction $_{\text{améliorée}}$) + (Fraction $_{\text{améliorée}}$/Accélération $_{\text{améliorée}}$))

  = 1/((1-F) + F/A))

Principes quantitatifs de conception des ordinateurs

Exemple :

Soit un dispositif d'amélioration 10 fois plus rapide que la machine de base, mais que l'on ne peut utiliser que 40% du temps.

Quelle est l'accélération apportée en introduisant ce dispositif ?

Principes quantitatifs de conception des ordinateurs

- $Fraction_{Am} = 0.4$
- $Accélération_{Am} = 10$

$$Accélération_{totale} = 1/(0.6 + (0.4/10)) = 1/0.61 \approx 1.56$$

La loi d'Amdahl :

- sert de guide pour calculer l'impact d'un dispositif sur les performances
- sert à comparer deux alternatives de conception

Principes quantitatifs de conception des ordinateurs

- **Equation de performance de l'UC**
  - **Temps UC** = Nbre cycles UC pour un programme x T
    = Nbre cycles UC x 1/f
    = NI x CPI x T
    = NI x CPI x I/f

    Mesurer les composantes de l'équation !
- **Localité des références**
  - Les programmes ont tendance à réutiliser les données et les instructions qu'ils ont utilisé récemment
  - Règle empirique : Un programme passe 90% de son temps d'exécution sur seulement 10% des instructions.
    La localité des références s'applique aussi aux accès aux données, mais moins fortement aux accès au code.

Principes quantitatifs de conception des ordinateurs

- **Assemblage**
  - Plus petit et plus rapide
  - Rendre le cas courant plus rapide

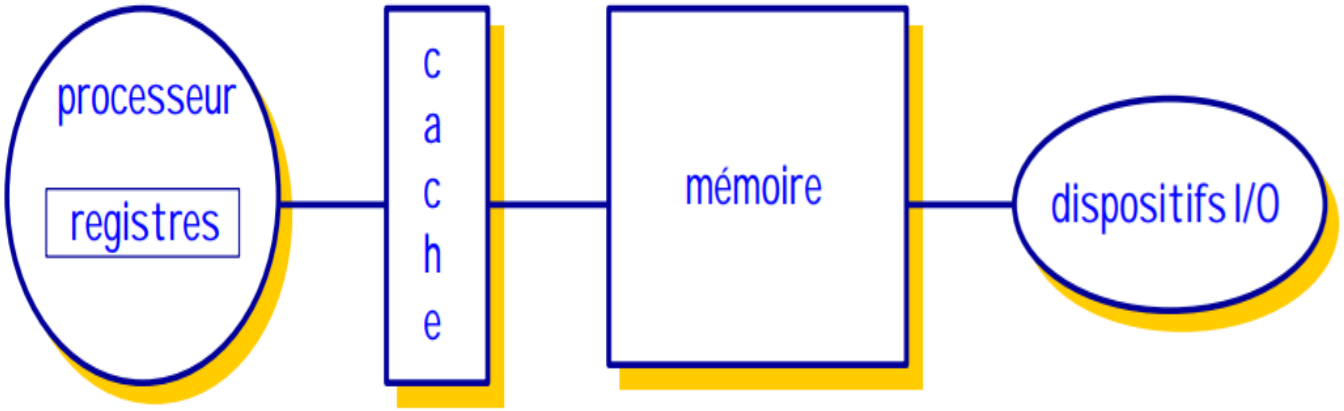
    ⇒ favoriser l'accès à de telles données améliorera les performances

    ⇒ les éléments auxquels on a accédé sont placés dans des mémoires plus rapides

Les petites mémoires sont plus rapides, on les utilise pour conserver les éléments les plus récemment utilisés près de l'UC

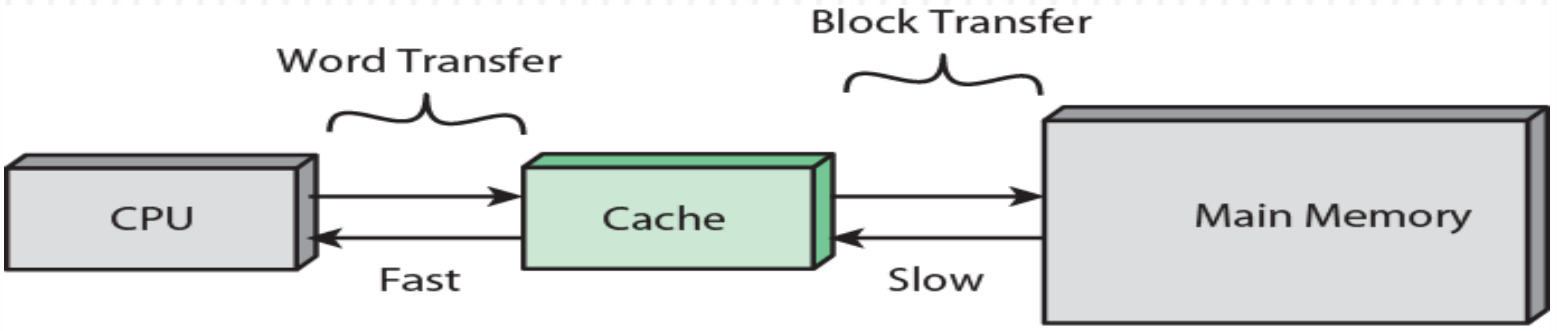
➡ ***Ces petites mémoires sont des caches***

Mémoire cache ?

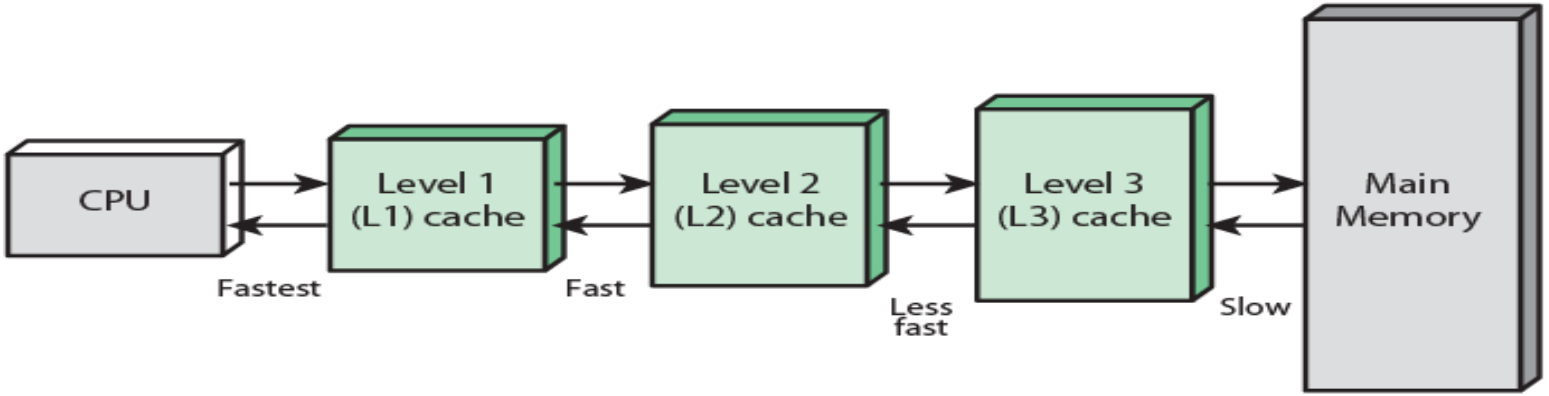


taille:	200B	64KB	32MB	2GB
vitesse:	5ns	10ns	100ns	5ms

Mémoires typiques



(a) Single cache



(b) Three-level cache organization

Propriété caches

- Exclusion Property
  - If block is in L1 cache, it is *never* in L2 cache
  - Saves some L2 space
- Inclusion Property
  - If block A is in L1 cache, it *must* also be in L2 cache

Mémoire cache

- Intégrée au processeur (mémoire cache de niveau 1) et proche de ce dernier (mémoire cache de niveaux 2 et 3), les mémoires cache sont des espaces offrant au processeur un accès rapide aux données et instructions les plus utiles. Elles lui épargnent des allers et venues incessants vers la mémoire principale (RAM).
- Compromis entre la taille et la vitesse de la mémoire
  - La mémoire cache est :
    - de petite capacité
    - de grande vitesse (facteur 10 entre cache et mémoire conventionnelle)
  - Réduire le temps d'accès moyen en conservant une mémoire importante

Mise en œuvre

- Elle est placée entre le processeur et la mémoire (Von Neumann)
- On espère que l'information soit souvent dans le cache
- Principe de la localité
  - L'accès est donc globalement plus rapide que dans le modèle VN
  - Même principe que la mémoire virtuelle

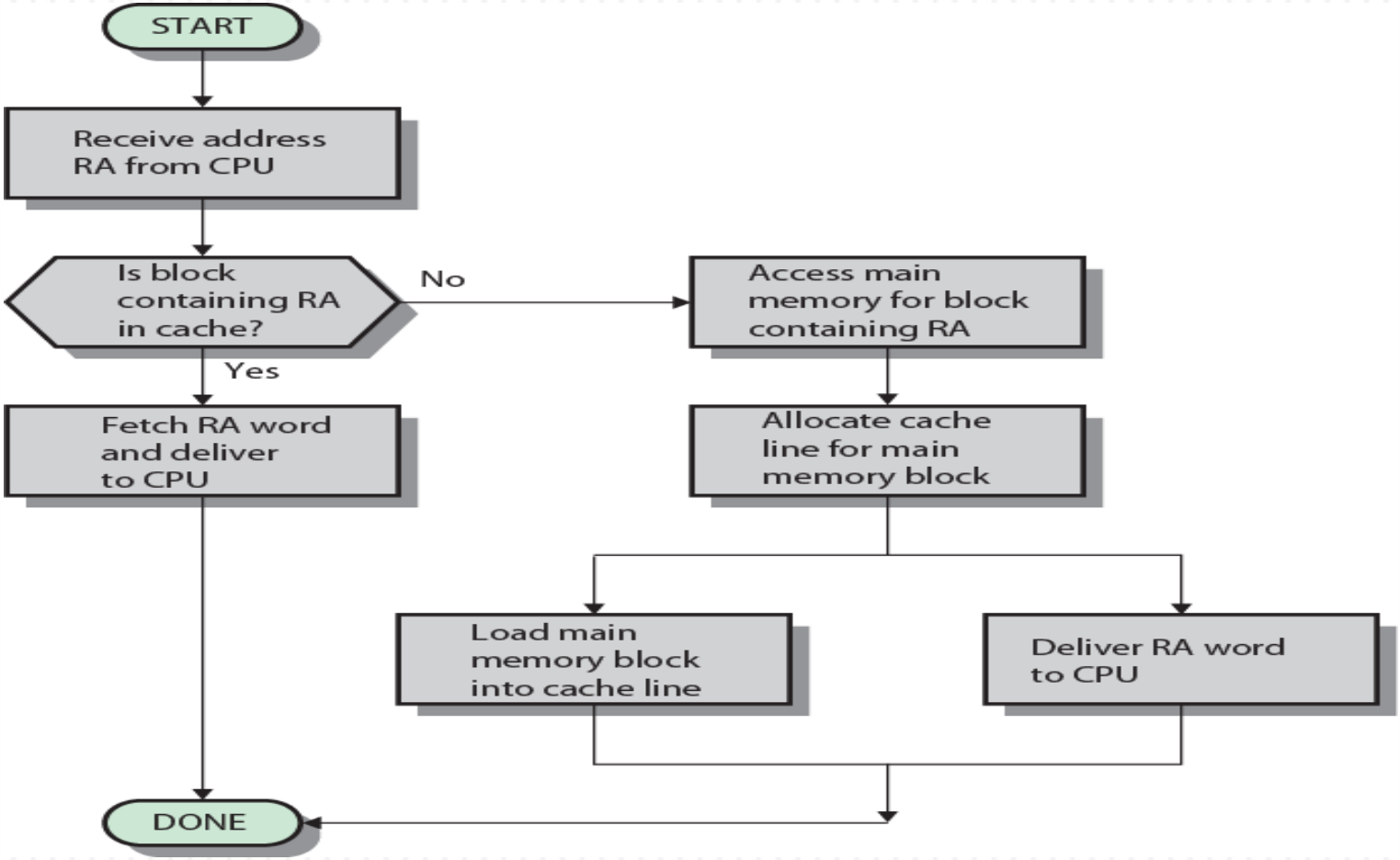
Localité des références

- **Localité spatiale :** le code d'un programme s'exécute toujours à l'intérieur de petites zones répétées de mémoire (des blocs correspondant à des boucles ou/et des sous-programmes)
- **En d'autres termes,** c'est la probabilité d'accès à une adresse voisine : Si un élément est référencé, les éléments dont les adresses sont voisines auront tendance à être référencés bientôt.

Localité des références

- **Localité temporelle :** les blocs s'exécutent en séquences très proches (il y a plus de chances d'accéder à une position de mémoire utilisée il y a 10 cycles qu'à une autre utilisée il y a 10000 cycles)
- **En d'autres termes,** probabilité d'accéder aux mêmes adresses successivement. Si un élément est référencé, il aura tendance à être référencé bientôt de nouveau.

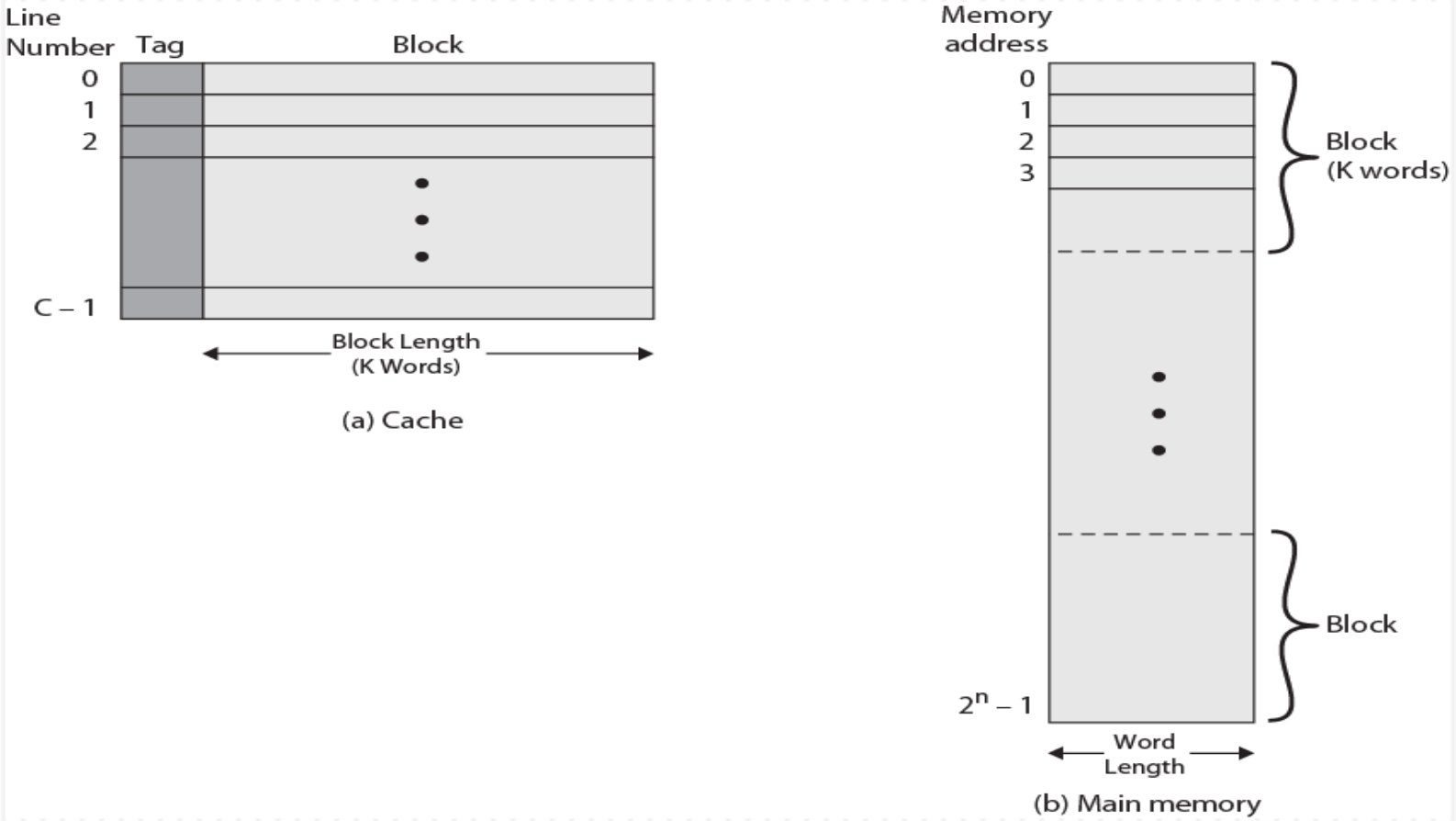
Algorithme du cache



Algorithme du cache

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Structure mémoire principale / cache



Conception de cache

- Addressing
- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Questions de conception de cache

- Où peut-on placer un bloc dans le niveau supérieur ? **Placement de bloc)**
- Comment trouver un bloc s'il est présent dans le niveau supérieur ?

  (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)

Placement des données

- Un élément crucial de l'efficacité du cache est de retrouver rapidement si des données à une adresse mémoire sont déjà dans le cache.
- Afin d'accélérer cette recherche, le nombre de lignes où peuvent être mises les données à une adresse mémoire fixée est souvent réduit. Ce nombre ne dépend pas de l'adresse et il est appelé le degré d'associativité du cache.
  - Lorsque ce nombre est réduit à 1, c'est-à-dire que les données de chaque adresse peuvent être mises dans une seule ligne du cache, on parle de cache direct.
  - Si au contraire l'associativité est égale au nombre de ligne du cache, c'est-à-dire que chaque donnée peut être mise dans n'importe quelle ligne du cache, le cache est dit complètement associatif.
  - Si l'associativité est un entier n, on parle de cache n-associatif (n-way set associatif en anglais).

Placement des données

- **Plusieurs stratégies :**
  - Direct mapping (Correspondance directe)
  - Fully associative mapping (espondance associative)
  - Set associative mapping ou n-way set assocoative (correspondance associative par ensemble)

Direct mapping

- Chaque donnée dans un seul emplacement (bloc) du cache
- Calcul simple :

i = j modulo m

Où :

i : numéro de l'emplacement du cache

j : adresse mémoire (numéro bloc)

m : nombre d'emplacements du cache

Table des lignes de cache

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
…	…
m-1	m-1, 2m-1,3m-1…2s-1

Direct mapping

- Les bits de poids faible de l'adresse dans le cache et dans la mémoire sont les mêmes
- Les bits de poids forts de l'adresse sont rangés dans le cache et divisés en deux champs :

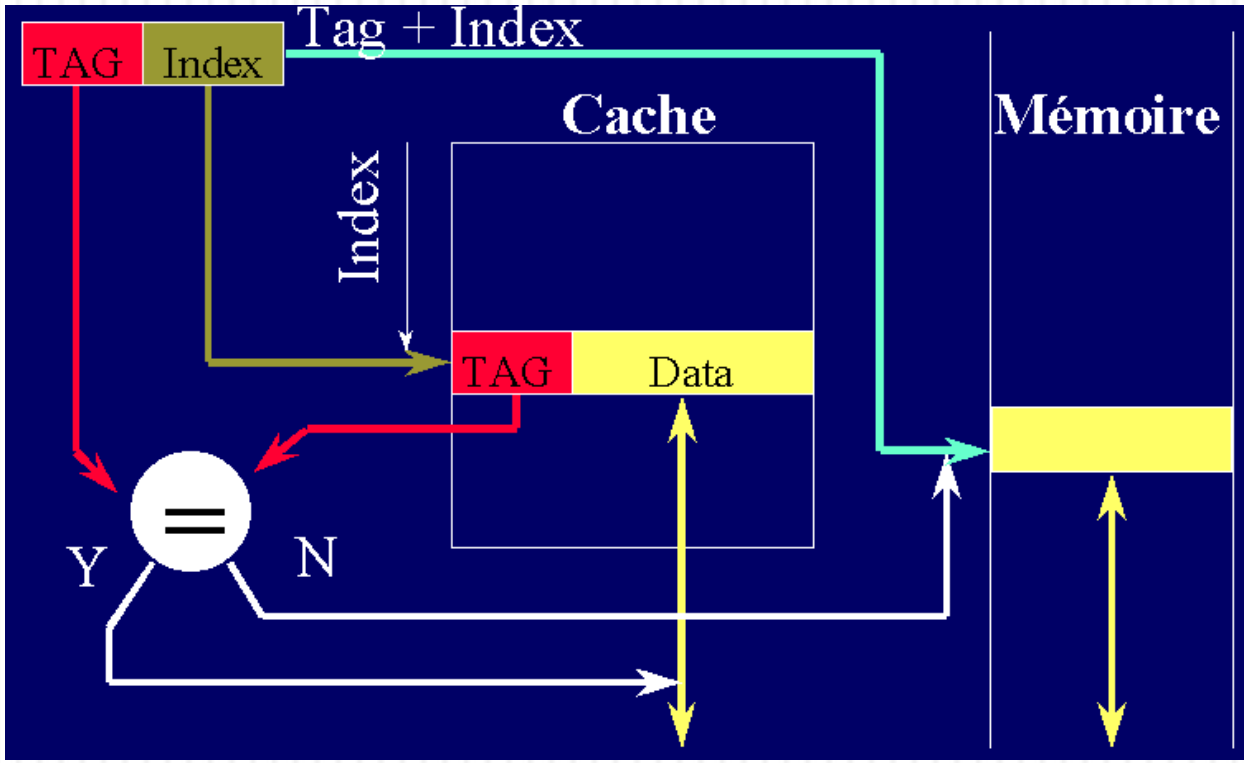
Méthode

- L'index permet l'accès à un emplacement du cache
- Un emplacement contient une étiquette + une donnée
- L'étiquette obtenue est comparée à l'étiquette de l'adresse. Si les deux étiquettes sont :
  - les mêmes ➡ Succès
  - différentes ➡ Echec

Accès mémoire

- En cas d'échec, l'adresse est envoyée à la mémoire :
  - **Lecture :** Le mot est envoyé au cache. Il peut être envoyé en simultané au processeur
  - **Ecriture :** Le mot est écrit dans le cache. Il peut être écrit en même temps dans la mémoire

Schéma



Accès mémoire

- En cas d'échec, on transfère toujours le bloc (ligne) qui contient le mot référencé :
  - Seul un bloc (ligne) du même numéro peut être présent dans le cache

Avantages / Inconvénients

- L'avantage des caches directs est de simplifier au maximum la recherche des données dans le cache.
  - Il suffit en effet de comparer l'étiquette de la ligne correspondante avec une partie de l'adresse.
  - Comme la ligne est unique, il est même possible de commencer la lecture du cache pendant la comparaison de l'étiquette avec l'adresse.
  - Si cette comparaison révèle un défaut de cache, cette lecture anticipée du cache est annulée.
- Les caches directs souffrent par contre d'un problème. Si un programme utilise simultanément deux parties de la mémoire qui doivent aller dans la même ligne de cache, il peut se produire de nombreux défauts de cache.

Avantages / Inconvénients

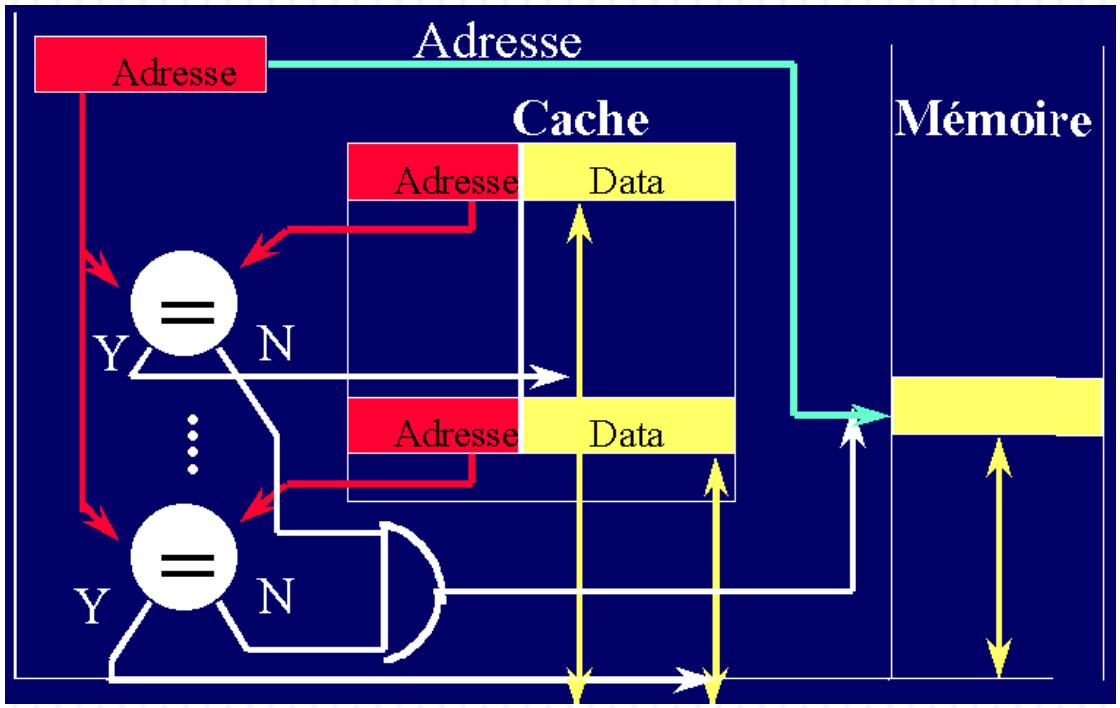
- **Avantages :**
  - Pas d'algorithme de remplacement
  - Matériel simple et peu coûteux
  - Rapide
- **Inconvénients :**
  - **Ratio ou taux de succès** faible
  - **Performances** décroissent si même index

Tendance *: Direct mapping adapté aux mémoires caches plus grandes*

Fully associative mapping

- L’adresse est simultanément comparée à toutes les adresses rangées dans le cache
- Un bit de validité est associé à chaque entrée
- Un algorithme de remplacement hardware

Fully associative mapping



Fully associative mapping

- **Avantages**
  - Flexibilité
  - Plusieurs lignes dans le cache avec le même numéro
- **Inconvénients**
  - Coût hardware des comparateurs
  - Algorithme de remplacement (lequel enlever ?)
  - Taille réduite

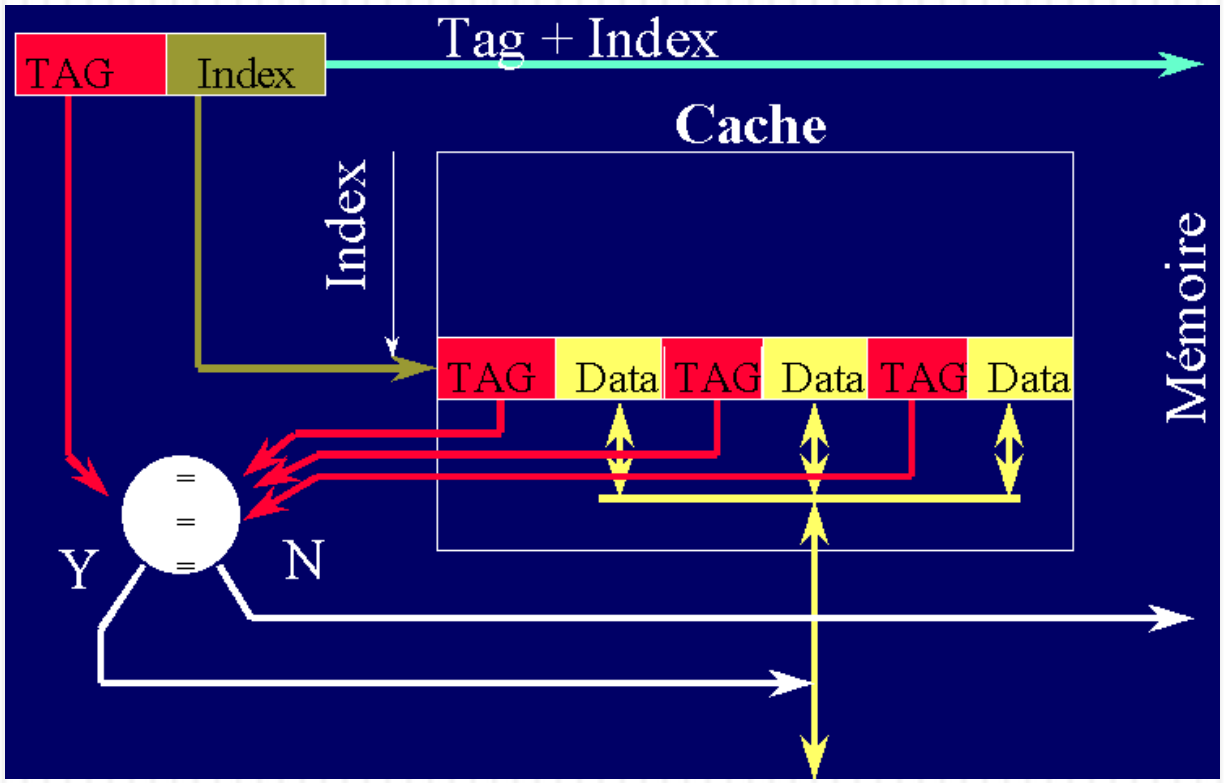
Set associative mapping

- Compromis entre direct et full associative mapping
- Permet un nombre limité de blocs (lignes) avec le même index mais des tag différents
- Le cache est divisé en « sets » de blocs. Chaque set contient le même nombre de blocs
- Chaque bloc a son propre Tag, qui associé à l'index, identifie le bloc

Set associative mapping

- Dans le cas des caches n-associatifs, l'organisation du cache est similaires aux caches directs.
  - À chaque adresse correspond n lignes consécutives du cache au lieu d'une.
  - L'indice de la première ligne est encore déterminé par des bits de poids faible de l'adresse.
  - On appelle généralement *ensemble* les blocs de mémoire ayant même indice.
- Pour n égal à 2 ou 4, un cache n-associatif se révèle presqu'aussi performant qu'un cache direct ayant 2 ou 4 fois plus de mémoire.
- Par contre, au delà de 8, le cache est pénalisé par le temps pris par la recherche des données dans le cache puisqu'il faut comparer une partie de l'adresse avec n étiquettes.
- On remarque en outre que les caches complètement associatifs ont des performances comparables aux caches 8-associatifs.

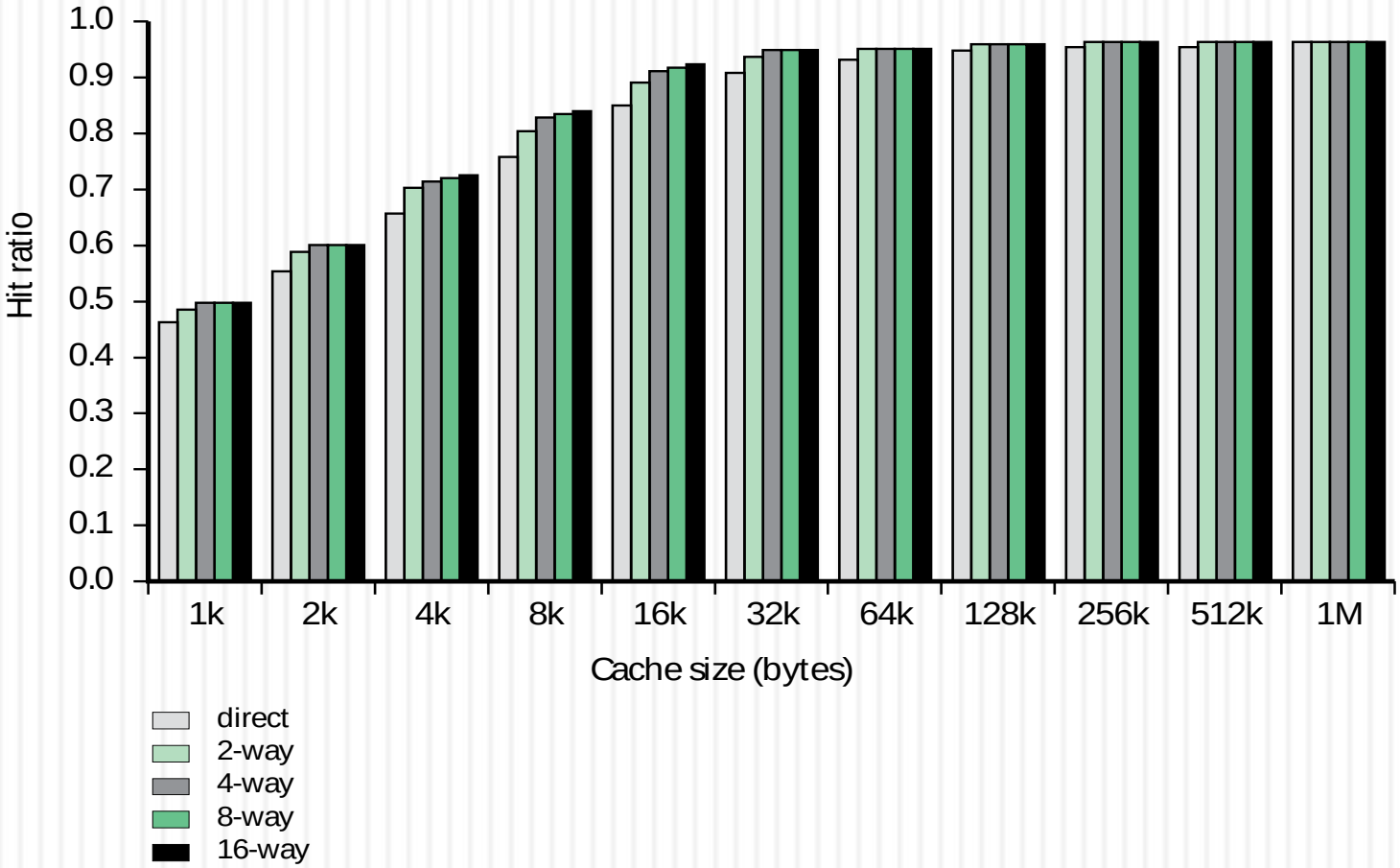
Set associative mapping



Set associative mapping

- **Avantages**
  - Moins de comparateurs
  - Algorithmes de remplacement seulement sur le set
  - Plus courant avec deux blocs par set

Taux de succès vs Associativité & taille



Comparaison de techniques

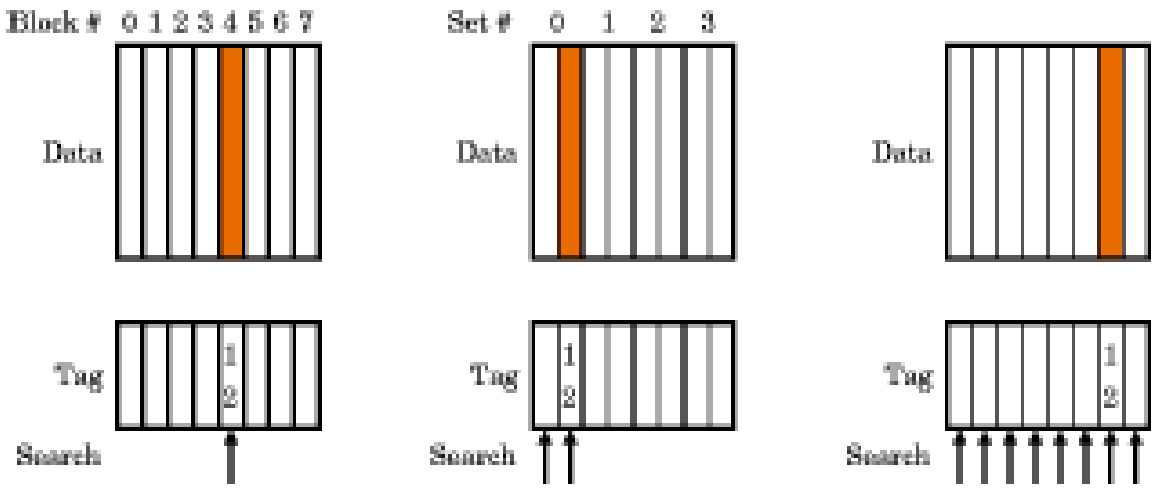
- **On hardware complexity :**
  - Fully associative cache requires special fast associative memory hardware
  - Direct mapping caches are much simpler in hardware terms
  - Set-associative caches offer a compromise

Comparaison de techniques

- **On usefulness**
  - direct mapping caches cannot normally cache blocks N, N+1 from main memory (since they would go into the same cache line)
    - This is a serious problem: many loops are bigger than one cache line, resulting in a cache miss (and cache reload) during the loop
    - This reduces the effectiveness of the cache
  - fully associative caches do not suffer from this problem at all (but are complex)
  - set-associative caches again proffer a compromise

Exercice 1

- Comment trouver un bloc (par exemple le bloc numéro 12) s'il est présent dans le niveau supérieur ?
  - Bit de validité
  - Relation adresse UC avec le cache



Politique de remplacement

- **Cache à correspondance directe**
  - 1 seule position possible dans le cache
  - pas de décision à prendre
- **Associatif par ensemble de N blocs**
  - Choix entre N positions possibles dans le cache
- **Totalement associatif**
  - Chaque case mémoire peut être placée n'importe

    où dans le cache

Politique de remplacement

- **Algorithmes de remplacement :**
  - Algorithme aléatoire
    - Le matériel choisit un bloc au hasard
  - LRU (Least Recently Used)
    - On remplace le bloc le moins récemment utilisé
    - Il faut garder l'historique des accès

Opérations d’écriture

- Consistance entre cache et mémoire

  (les deux versions de la donnée peuvent être différentes)
- Nécessité de conserver la cohérence
  - dans les systèmes multiprocesseurs (mémoire partagée avec caches multiples)
  - ou avec des entrées-sorties sur la mémoire

Write through (Ecriture simultanée)

- Chaque écriture dans le cache est répétée sur la mémoire.
- Le cache est plus efficace alors en lecture qu'en écriture
- En moyenne (Smith 1982) 3 à 10 lectures entre 2 écritures.

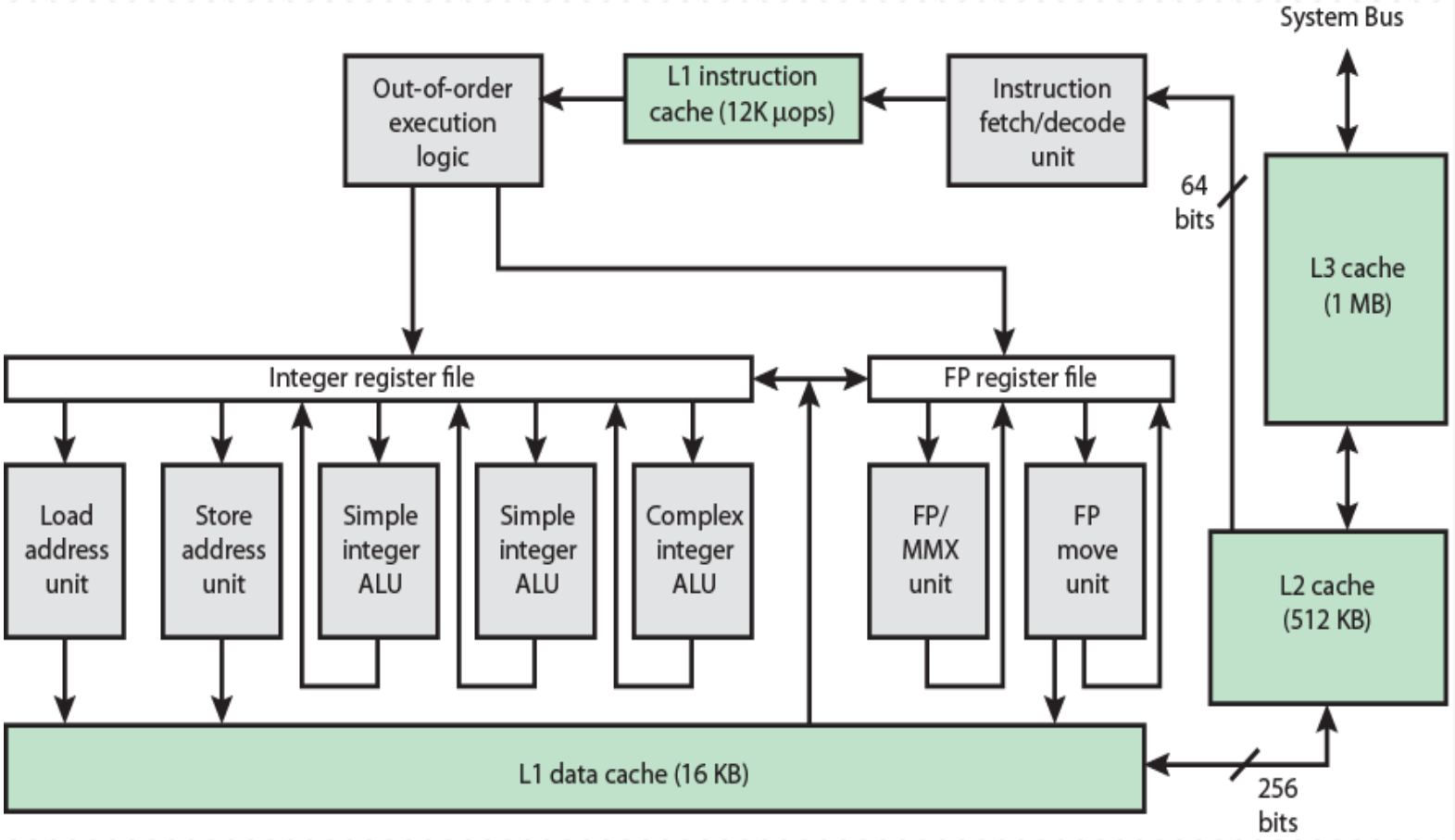
Write back (Réécriture)

- L’écriture n’est réalisée que lors du remplacement du bloc
- Un bit de modification est nécessaire

Exemple

- **Caches du pentium 4**
  - **Cache de niveau 1 de 8 Ko associatif par ensemble de 4 blocs de 64 octets chacun, à écriture simultanée**
  - **Cache d'instructions (µops) de 12 Ko**
  - **Cache unifié de niveau 2 de 256 Ko, associatif par ensemble de 8 blocs de 128 octets chacun, à réécriture.**

Caches Pentium 4



Tailles des caches

Processor	Type	Year of Introduction	L1 cache	L2 cache	L3 cache
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTAb	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/server	2002	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1MB	—

Sources de défauts cache

- **Obligatoire**
  - première référence (démarrage à froid)
    - On ne peut rien y faire
- **Conflit / Collision**
  - Plusieurs cases mémoire sur la même position
    - Solution 1 : augmenter la taille du cache
    - Solution 2 : augmenter l'associativité
- **Capacité**
  - Le cache ne peut contenir tous les blocs accédés par un programme
    - Solution : augmenter la taille du cache

Exercice 2

Etant donné une mémoire cache de 2048 mots et une mémoire principale de 32 kmots. Une ligne comporte 16 mots. X et Y sont des tableaux de mots se trouvant en mémoire aux adresses 0x2000 et 0x3000 respectivement. Compléter le tableau pour la séquence ci-dessous en indiquant les succès et les défauts cache (préciser le type de défaut) :

```
for (i=0 ;   i<N ;  i++) ;   s = s + X[i] * Y[i] ;
```

Numéro Itération	Correspondance directe		2-way set associative	
	X	Y	X	Y
0				
1-15				
16				

Exercice 2 (Réponse)

- Taille mémoire principale = 32 Kmots = 2048 blocs, longueur @=15 bits
- Taille cache = 2048 mots = 128 blocs donc 7 bits pour index
- Taille ligne (ou bloc) = 16 mots, 4 bits pour offset
- Adresse tableau X : 0x2000
- Adresse tableau Y : 0x3000
- Organisation cache à correspondance directe
- **Itération 0 :**

Tag (4 bits)	Index (7 bits)	Offset (4 bits)
0100	0000000	0000

bloc référencé est : 0 (indiqué par la valeur de l'index)

Exercice 2 (Réponse)

L'élément X[0] référencé, provoque un défaut de type obligatoire, cela entraine le chargement de la mémoire principale du bloc contenant l'élément X[0]

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]	X[11]	X[12]	X[13]	X[14]	X[15]

L'élément Y[0] référencé provoque aussi un défaut de type obligatoire, cela entraine le chargement de la mémoire principale du bloc contenant l'élément Y[0]

4	7	4 bits
0110	0000000	0000

Y[0]	Y[1]	Y[2]	Y[3]	Y[4]	Y[5]	Y[6]	Y[7]	Y[8]	Y[9]	Y[10]	Y[11]	Y[12]	Y[13]	Y[14]	Y[15]

Exercice 2 (Réponse)

- **Itérations 1 à 15**
  - X[1], son bloc était dans le cache puis évincé en MP à cause de Y[1] donc défaut de type conflit, même chose pour les autres éléments X[2] … X[15]
  - Y[1] à Y[15] : provoquent aussi des défauts de type conflit
- **Itération 16**
  - X[16] et Y[16] provoquent chacun à son tour un défaut de type obligatoire.
    - Le bloc contenant X[16] dans la mémoire principale ira à l'emplacement (bloc) cache 0.
    - De même Y[16], ira loger dans l'emplacement 0, en remplacement du bloc contenant X[16].

Exercice 2 (Réponse)

- **2-way set associative**
  - **Itération 0**
    - 128 blocs/2 = 64 ensembles donc 6 bits pour Index
    - Adresse sur 3 champs

5	6	4 bits
01000	000000	0000

    - Après le traitement du défaut de l'élément X[0], on obtient :

Set O

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]	X[11]	X[12]	X[13]	X[14]	X[15]

    - Après le traitement de l'échec de l'élément Y[0], on obtient :

Set O

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]	X[11]	X[12]	X[13]	X[14]	X[15]
Y[0]	Y[1]	Y[2]	Y[3]	Y[4]	Y[5]	Y[6]	Y[7]	Y[8]	Y[9]	Y[10]	Y[11]	XY12]	Y[13]	Y[14]	Y[15]

Exercice 2 (Réponse)

- **2-way set associative (suite)**
  - **Itérations 1 à 15**
    - Les éléments X[1] et Y[1] réalisent chacun un succès
    - X[2] et Y[2] … X[15] et Y[15] réalisent également des succès
  - **Itération 16**
    - X[16] et Y[16] provoquent respectivement un défaut de type obligatoire et iront loger dans les deux blocs du set 0 (selon algorithme de remplacement)

Exercice 2 (Réponse)

- **Synthèse :**

	Correspondance directe		2-way set associative	
Itération	X	Y	X	Y
0	obligatoire	obligatoire	obligatoire	obligatoire
1-15	conflit	conflit	succès	succès
16	obligatoire	obligatoire	obligatoire	obligatoire

Performances des caches

- **Temps d'accès moyen** = temps d'accès$_{réussi}$ + taux d'échec x pénalité échec$_{unifiés\ et\ séparés}$)
- **Temps UC** = NI x (CPI$_{exécution}$ + cycles d'attente mémoire/instruction) x temps de cycle
  - **Cycles d'attente mémoire** = Nombre de lectures x x Taux d'échec $_{lecture}$ x Pénalité échec $_{lecture}$ + Nombre d'écritures x Taux d'échec $_{écriture}$ x Pénalité échec $_{écriture}$

Exercice 3

On suppose que la pénalité d'échec est de 50 cycles et toutes les instructions prennent normalement 2.0 cycles . On suppose que le taux d'échec est de 2 % et qu'il y a une moyenne de 1.33 références mémoire par instruction.

Quel est l'impact sur la performance quand on considère le comportement du cache ?

Exercice 3 (Réponse)

On suppose que la pénalité d'échec est de 50 cycles et toutes les instructions prennent normalement 2.0 cycles . On suppose que le taux d'échec est de 2 % et qu'il y a une moyenne de 1.33 références mémoire par instruction. Quel est l'impact sur la performance quand on considère le comportement du cache ?

temps UC = NI x ($CPI_{exécution}$ + cycles d'attente mémoire/instruction) x temps de cycle

La performance en tenant compte des défauts cache :

Temps UC = NI x (2.0 + 1.33 x 2 % x 50) x temps de cycle

= NI x 3.33 x temps de cycle

Le temps de cycle et NI ne changent pas avec ou sans cache. Temps UC croit avec le CPI

Aspects cache

Transparency

- the cache should not be visible to the programmer at all
- it should take advantage of general characteristics of programs
  - not require programs to be specially designed to take advantage of it
  - this does mean that it is possible to write cache-defeating programs!

Hit Ratio

- If a memory access finds the datum in the cache, this is a *hit*
- if not, this is a *miss*
- the hit ratio is defined to be

$$\frac{\text{Hits}}{\text{Hits + Misses}}$$

Clearly, high hit ratios (near one) are desirable.


Thank's

QCM

- Quelle affirmation est fausse ?
  - La RAM stocke des informations de manière temporaire
  - La ROM peut être lue, mais son contenu ne peut être effacé
  - L'extinction de la machine efface tout le contenu de la mémoire cache
  - L'extinction de la machine efface le contenu de toutes les mémoires de l'ordinateur
- La mémoire cache d'un ordinateur permet :
  - une plus grande sécurité des données
  - un accès rapide aux données
  - une compression des données

QCM

- Quelle est la principale caractéristique d'un circuit EPROM ?
  - Il est effaçable électriquement
  - Il peut être programmé plusieurs fois
  - Il fait partie de la famille des mémoires vives
- Quelle est la principale caractéristique d'un circuit EEPROM ?
  - Il est effaçable électriquement
  - Il peut être programmé plusieurs fois
  - Il est effaçable par rayonnement

QCM

- La mémoire cache est :
  - Une mémoire que l'on ne peut pas voir
  - Une mémoire de grande capacité
  - Une mémoire à accès rapide
  - Une mémoire en lecture seule
- Lequel de ces types de mémoire est le plus rapide ?
  - registre
  - disque optique
  - cache
  - mémoire principale

QCM

- Une puce informatique qui garde son contenu indéfiniment sans rafraîchissement électronique constant, est
  - RAM dynamique (DRAM)
  - ROM dynamique
  - RAM statique (SRAM)
  - ROM statique
- Les caches peuvent également être insérés entre la mémoire principale et la mémoire de masse.
  - Vrai
  - Faux

QCM

- Quelle description décrit le mieux le concept de localité spatiale ? Lorsqu'une adresse mémoire est référencée,
  - elle tend à être référencée de nouveau prochainement
  - elle tend à être référencée de nouveau après un certain temps
  - les adresses situées à proximité tendent à être référencées
  - les adresses situées à proximité sont rarement référencées
  - elle ne sera plus référencée par la suite
  - les adresses situées à proximité ne seront pas référencées

QCM

- Quelle description décrit le mieux le concept de localité temporelle ? Lorsqu'une adresse mémoire est référencée,
  - elle tend à être référencée de nouveau prochainement
  - elle tend à être référencée de nouveau après un certain temps
  - les adresses situées à proximité tendent à être référencées
  - les adresses situées à proximité sont rarement référencées
  - elle ne sera plus référencée par la suite
  - les adresses situées à proximité ne seront pas référencées

QCM ?

- Quel type de cache est le mieux adapté pour un programme effectuant beaucoup de lectures séquentielles dans un énorme tableau ?
  - Direct mapped cache avec des blocs de 8 mots,
  - Direct mapped cache avec des blocs d'un mot,
  - Fully associative cache avec des blocs d'un mot,
  - Fully associative cache avec des blocs de 4 mots,
  - Aucune de ces réponses

QCM ?

- Quel type de cache est le mieux adapté pour un programme effectuant beaucoup d'écriture à répétition dans des emplacements aléatoires en mémoire ?
  - Direct mapped-cache avec write-back,
  - Direct mapped cache avec write-through,
  - Set-associative cache avec write-back,
  - Set-associative cache avec write-through,
  - Aucune de ces réponses