Mémoire: optimisations

GIF-1001 Ordinateurs: Structure et Applications, Hiver 2016

Jean-François Lalonde

Mémoire

- Un emplacement qui contient:
  - les données manipulées par le microprocesseur
  - les instructions (programmes) à exécuter par le microprocesseur
- Analogie: boîtes aux lettres
  - adresse (indique quelle boîte aux lettres choisir)
  - contenu (la lettre qu'il y a dedans)

Mémoire

- Un mot de mémoire se retrouve à chaque adresse. Les mots sont constitués de plusieurs bits
- On décrit une mémoire grâce à deux chiffres (indépendants):
  - le nombre d'adresses possibles (ici: 2^{16} = 65,536 adresses)
  - la taille des mots de la mémoire (ici: 8 bits = 1 octet)
- Les mémoires qui peuvent se lire et s'écrire possèdent au moins trois signaux de contrôle du microprocesseur:
  - Lecture de la mémoire;
  - Écriture de la mémoire;
  - Activation (Enable) de la mémoire.

mémoire de 2^{16} adresses

Adresse	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0x0000								
0x0001								
0x0002								
0x0003								
0x0004								
0x0005								
0x0006								
0x0007								
0x0008								
0x0009								
0x0010								
0x0011								
0x0012								
0x0013								
0x0014								
0x0015								
…								
…								
0xFFFF								

taille des mots = 8 bits = 1 octet

Types de mémoires

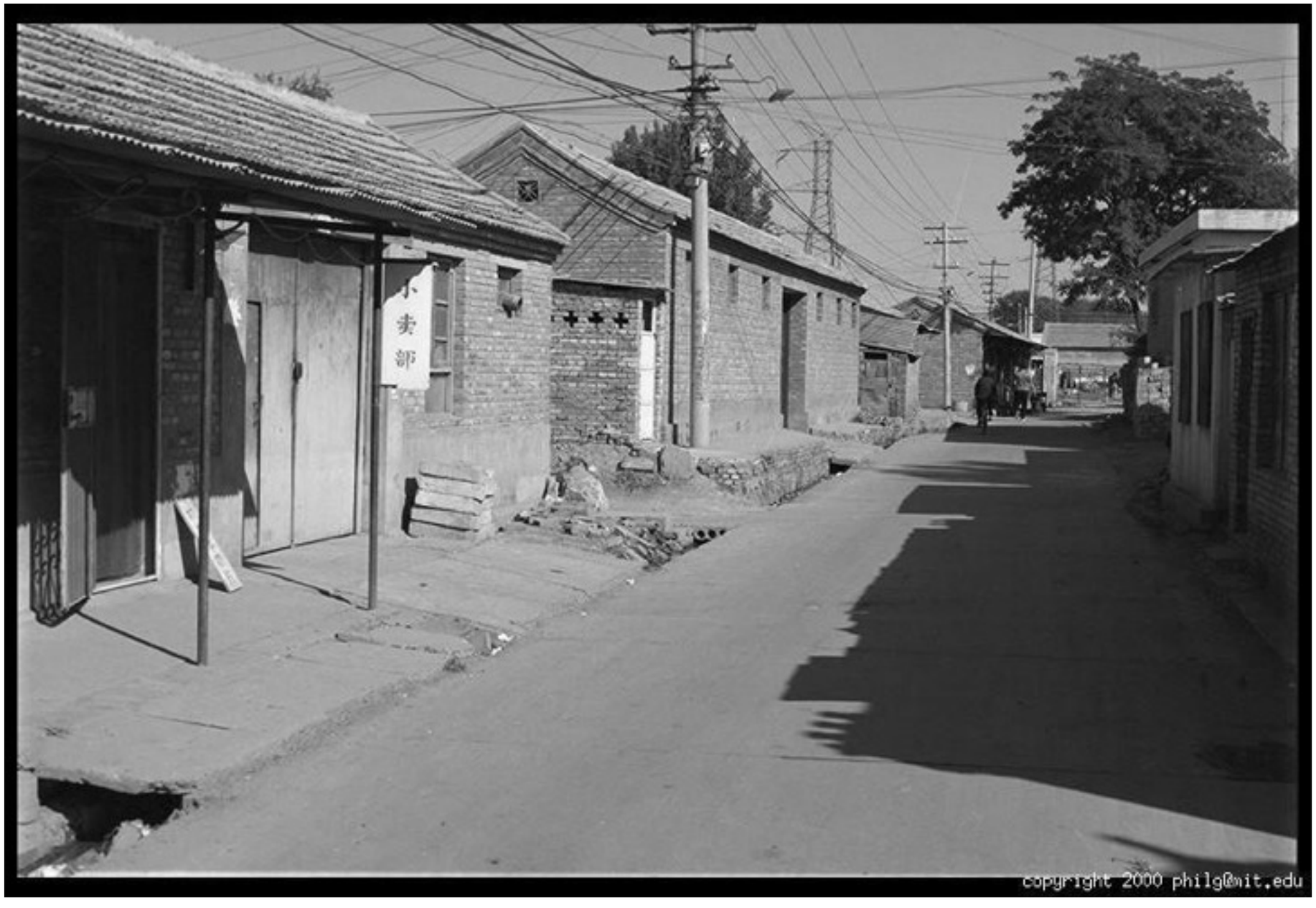
- Les mémoires peuvent être:
  - volatiles: perdent leur contenu lorsqu'elles perdent leur alimentation;
  - ou non-volatiles: conservent leur contenu même sans alimentation.
- Les mémoires volatiles peuvent être:
  - statiques: n'ont pas besoin d'être lues pour conserver leurs valeurs
  - dynamiques: nécessitent un rafraîchissement de leur données de façon périodique. Si les données d'une mémoire dynamique ne sont pas "lues" régulièrement, elles s'effacent.
- Les mémoires
  - ROM: ne peuvent pas être écrites (Read Only Memory)
  - RAM: peuvent être écrites (Random Access Memory);
- Les noms sont donnés aux mémoires en fonction de ces caractéristiques. Par exemple, SRAM est de la RAM Statique.

Types de mémoire

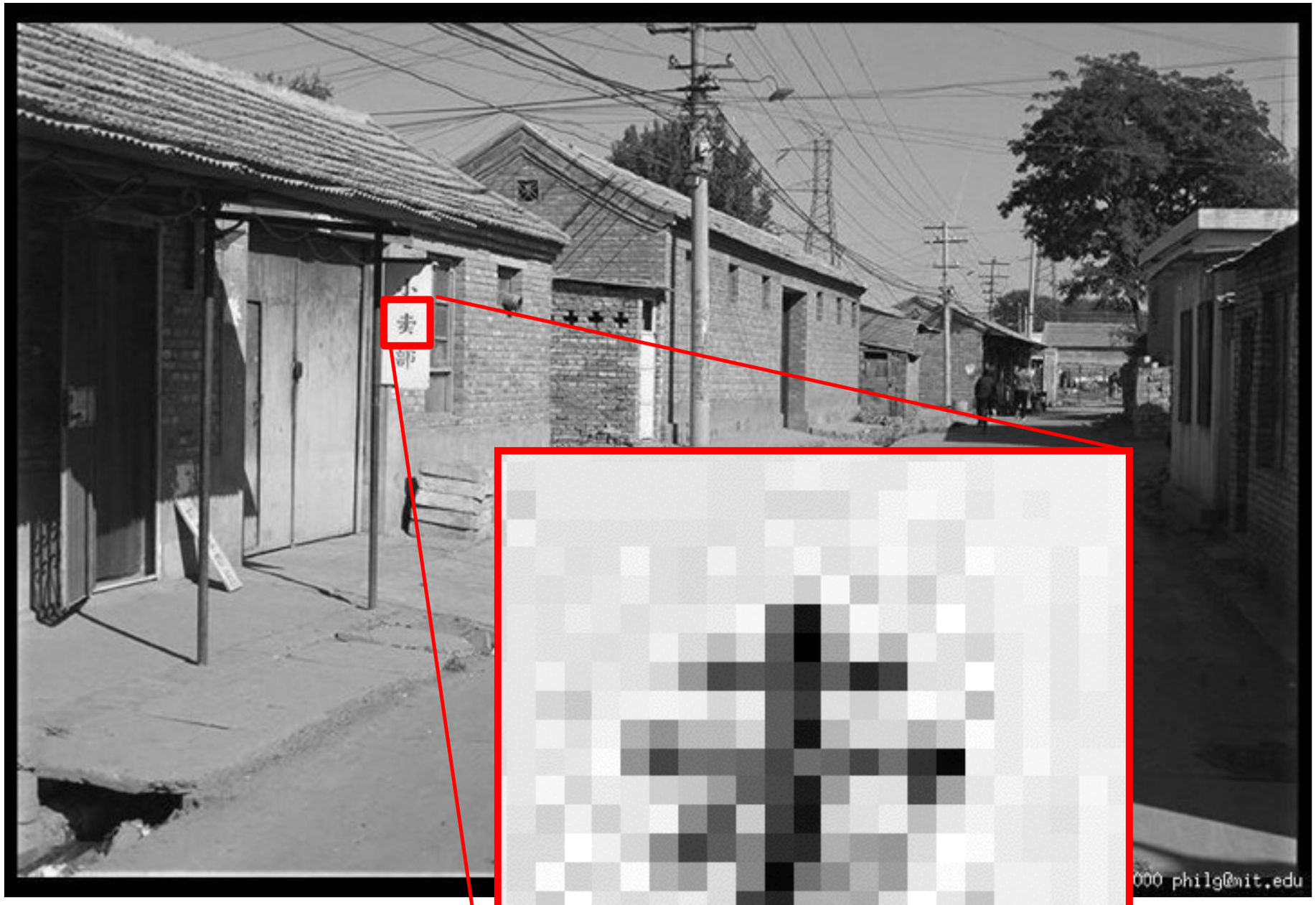
Noms	Volatile?	Dynamique?	ROM?	Coût
SRAM	oui	non	non	cher
DRAM	oui	oui	non	moyen
ROM	non	non	oui	faible
UVPROM	non	non	oui, mais peut être reprogrammée par UV	moyen
EEPROM	non	non	oui, mais peut être reprogrammée électroniquement	moyen
Flash	non	non	oui, mais peut être reprogrammée électroniquement	cher
Magnétique	non	non	non	faible

- Certaines mémoires peuvent s'effacer (et être réécrites) à l'aide d'une procédure spéciale:
  - UVPROM nécessite de l'ultra-violet (plus utilisé)
  - Flash et EEPROM s'effacent électriquement mais avec une opération spéciale
- Autres types de mémoires:
  - SDRAM (Synchronous DRAM): similaire à DRAM
  - DDR (Double Data Rate): SDRAM, mais deux fois plus rapide, DDR2 deux fois plus rapide que DDR, … DDR4 (2014, maximum de 128GB sur une barrette)
  - …

Une matrice de pixels



Une matrice de pixels



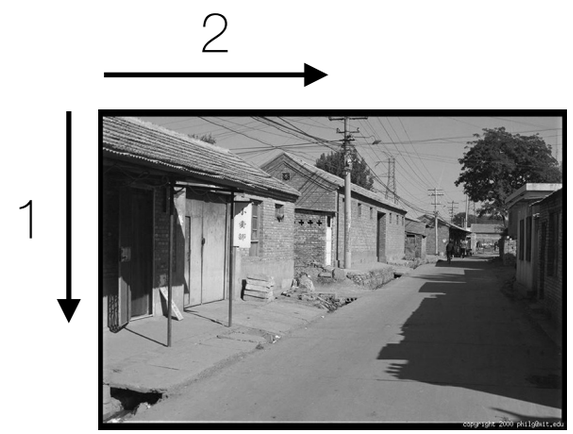
Une matrice de pixels

235	237	240	247	158	94	217	247	237	235	252
242	227	209	227	143	79	191	235	207	242	232
227	184	130	140	130	107	145	105	125	232	235
245	242	224	240	143	117	232	222	230	247	242
181	207	207	222	145	94	204	224	227	201	217
125	158	153	148	128	153	148	128	156	115	84
219	214	189	148	130	99	186	235	232	125	189
245	171	138	217	122	94	224	230	240	209	237
176	125	143	168	110	107	196	186	181	230	252
201	186	230	171	84	156	176	201	186	237	247
232	240	227	125	105	199	199	196	227	252	237

Aujourd'hui

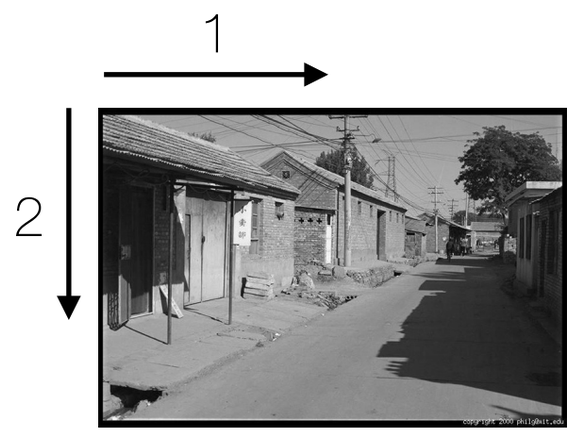
- Laquelle de ces deux versions est la plus rapide?

```
for (int i = 0; i < 256; i++) {
  for (int j = 0; j < 512; j++) {
    img[i*512 + j] = 0;
  }
}
```



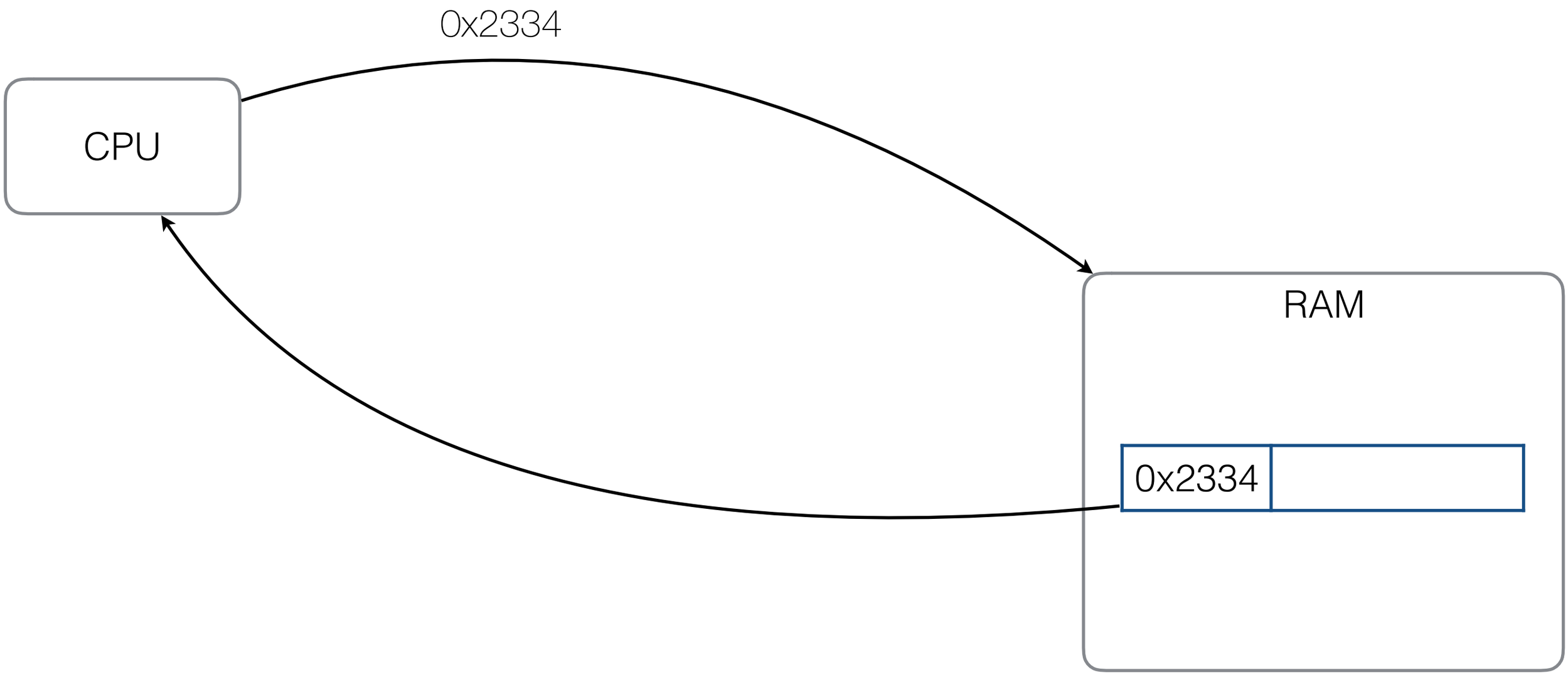
ou

```
for (int j = 0; j < 512; i++) {
  for (int i = 0; i < 256; j++) {
    img[i*512 + j] = 0;
  }
}
```

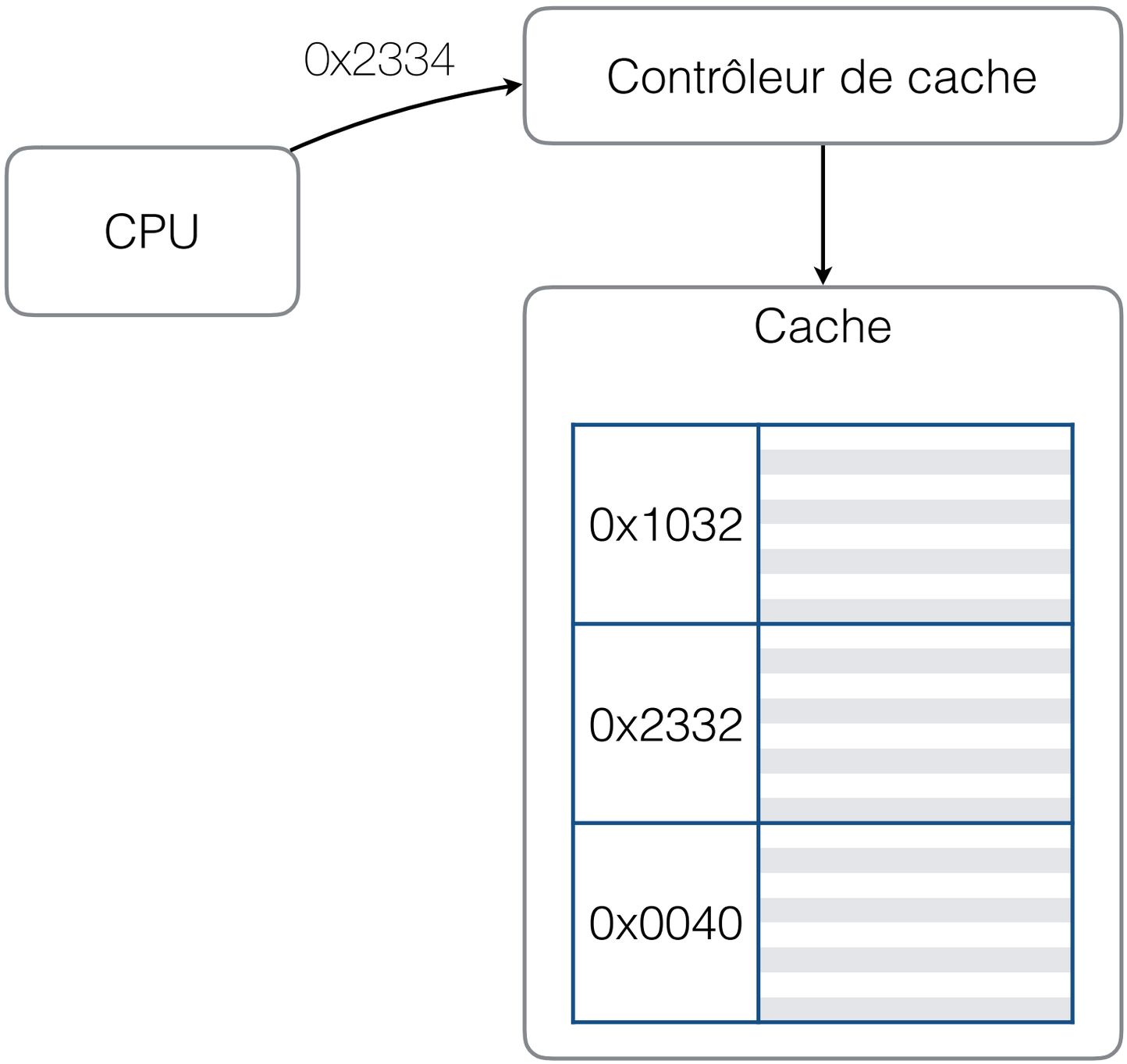


i = lignes ("y")
j = colonnes ("x")

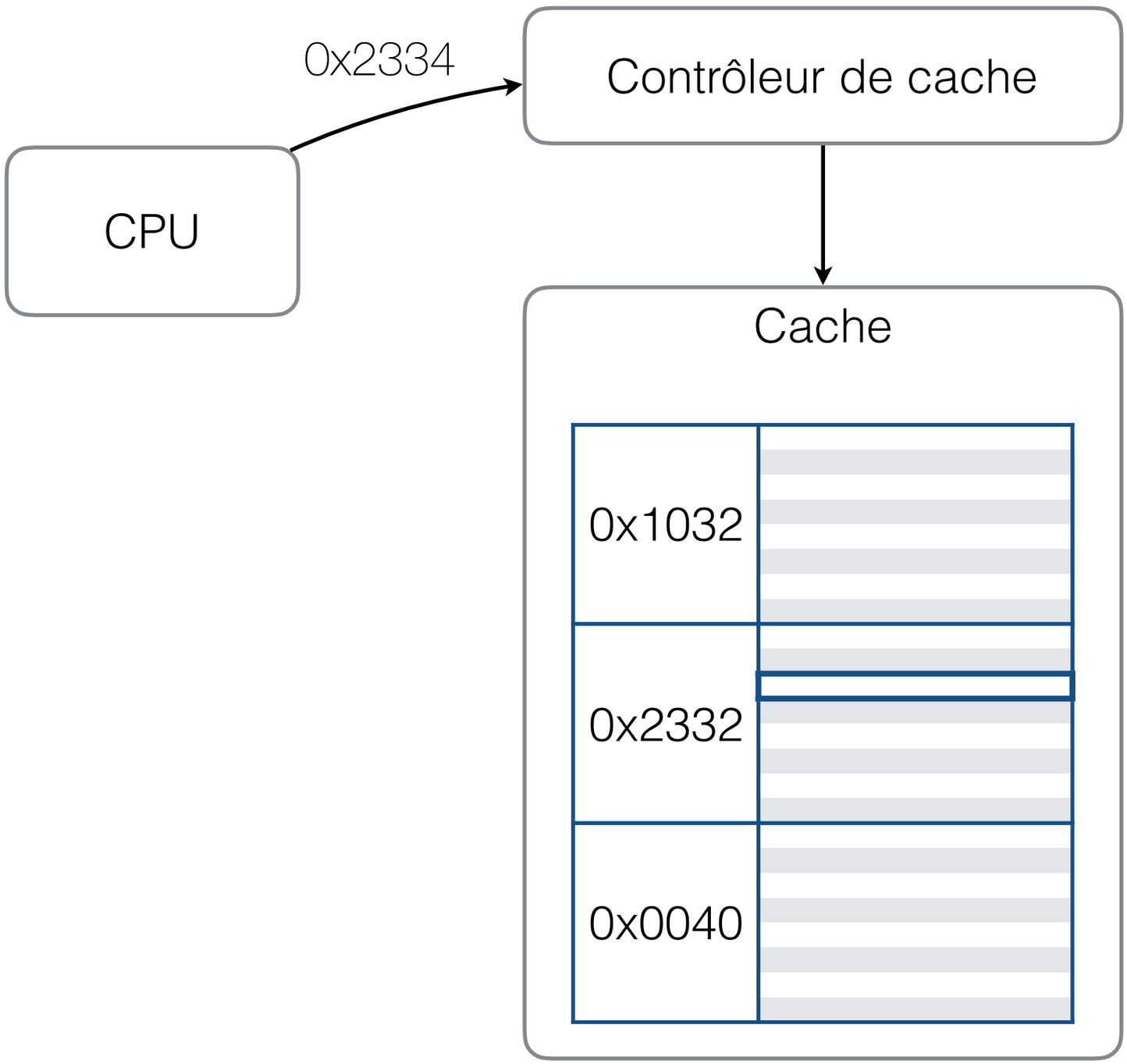
Lecture en mémoire



On rajoute une cache!

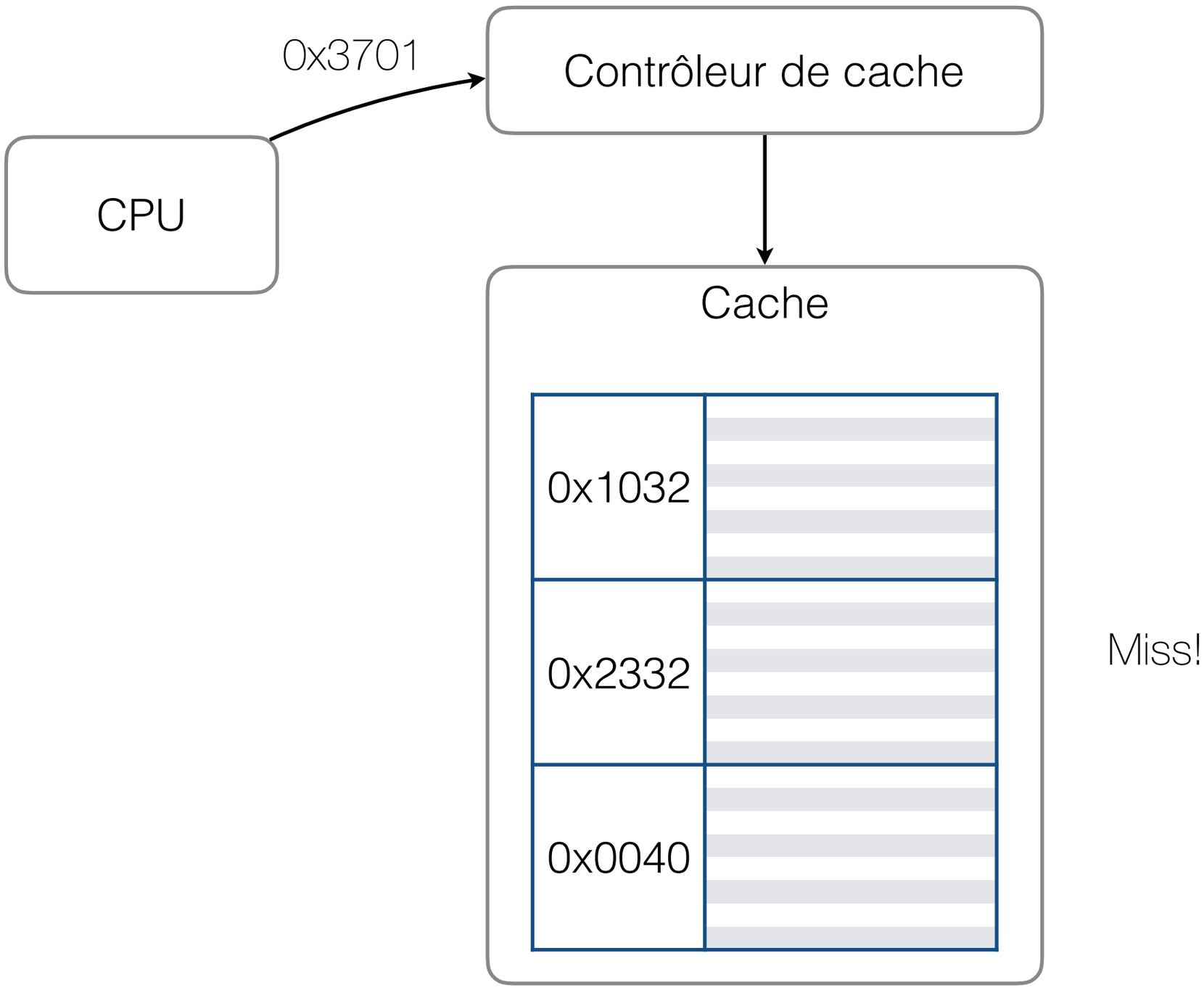


Lecture en cache



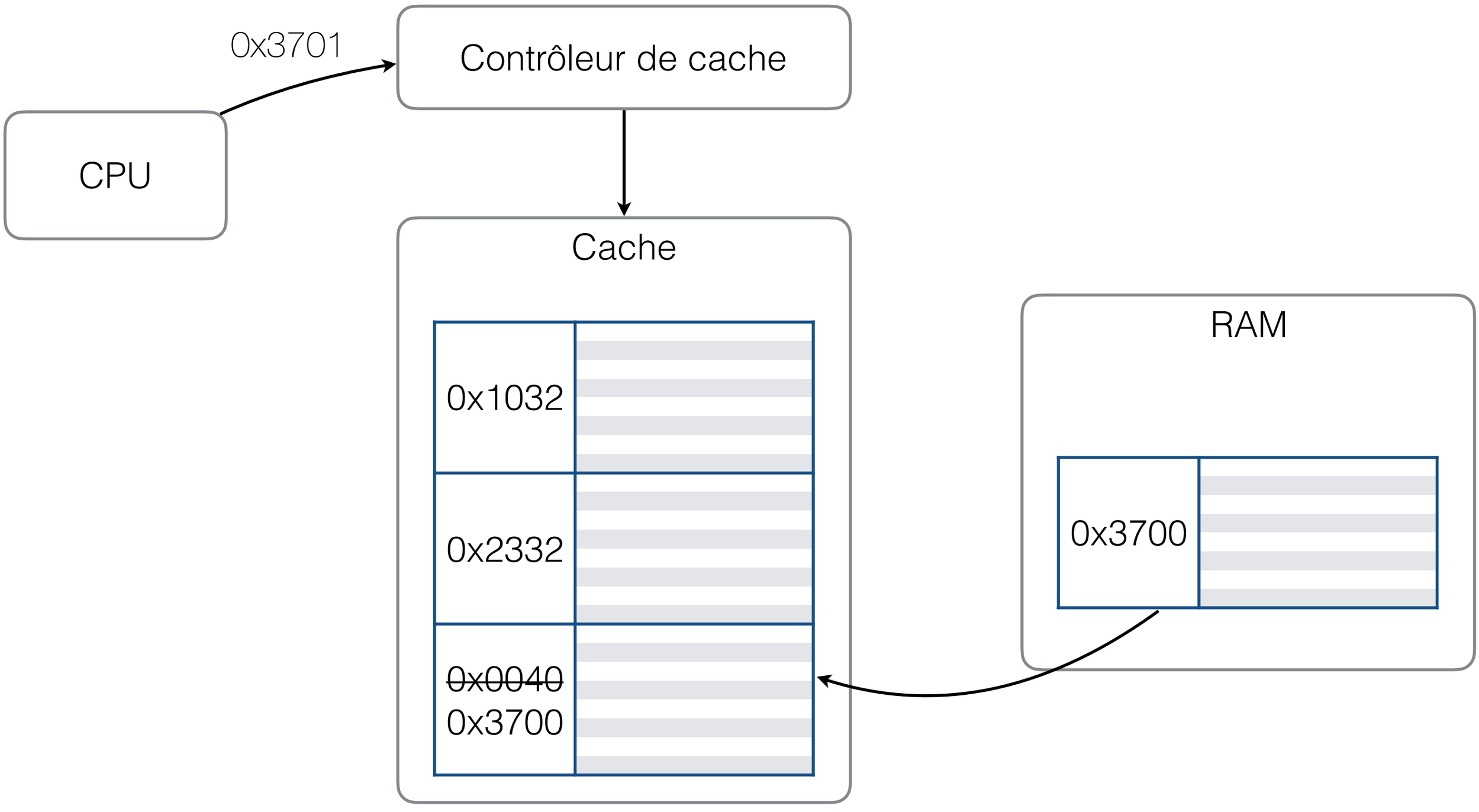
Hit!

Lecture en cache



Miss!

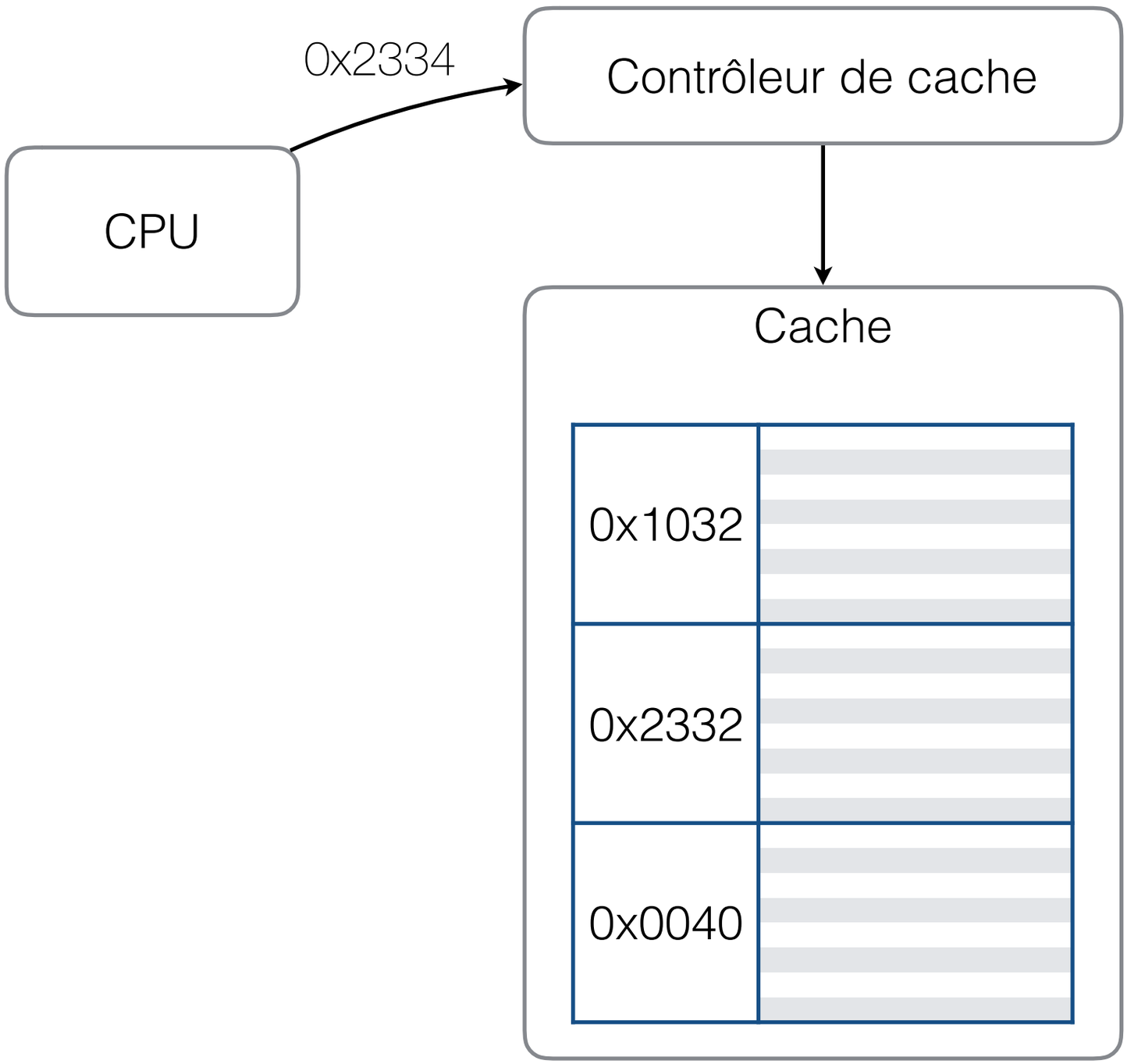
Lecture en cache



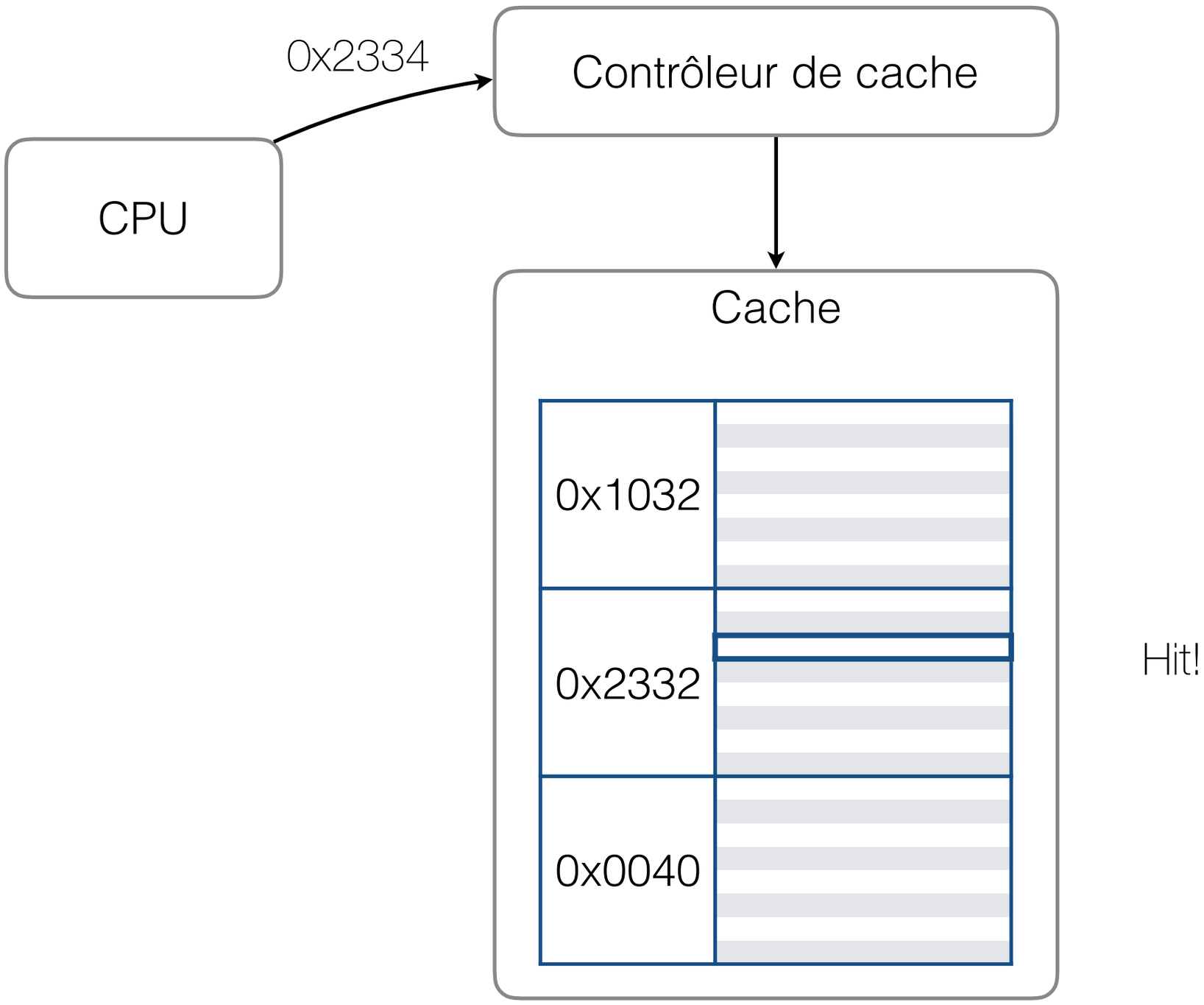
2 stratégies

- Que faire lorsque l'on veut écrire en cache?
  - « **Write-through** » : écrire les changements dans la RAM au fur et à mesure
    - L'exemple précédent utilisait cette technique

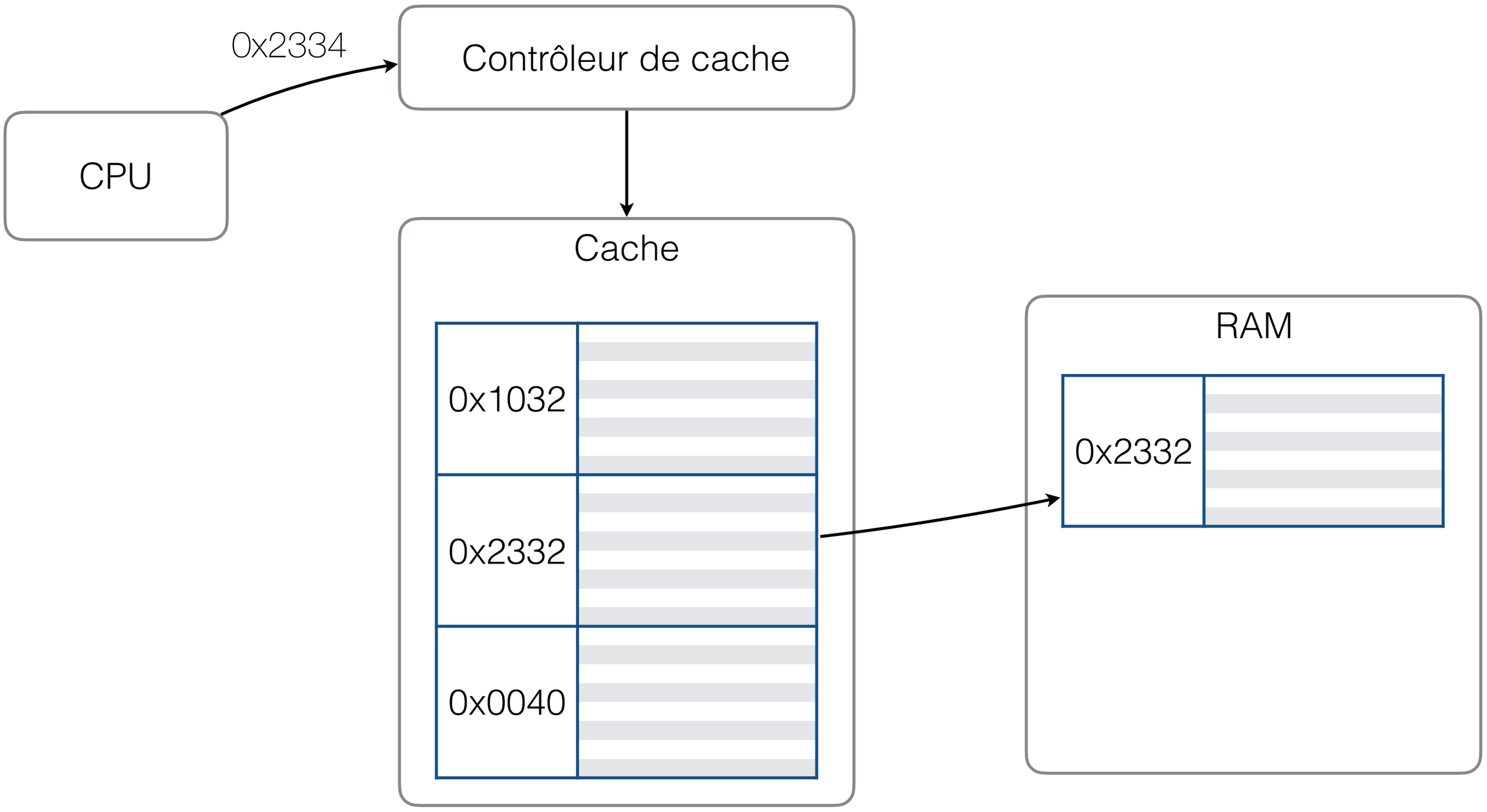
Écriture en cache « write-through »



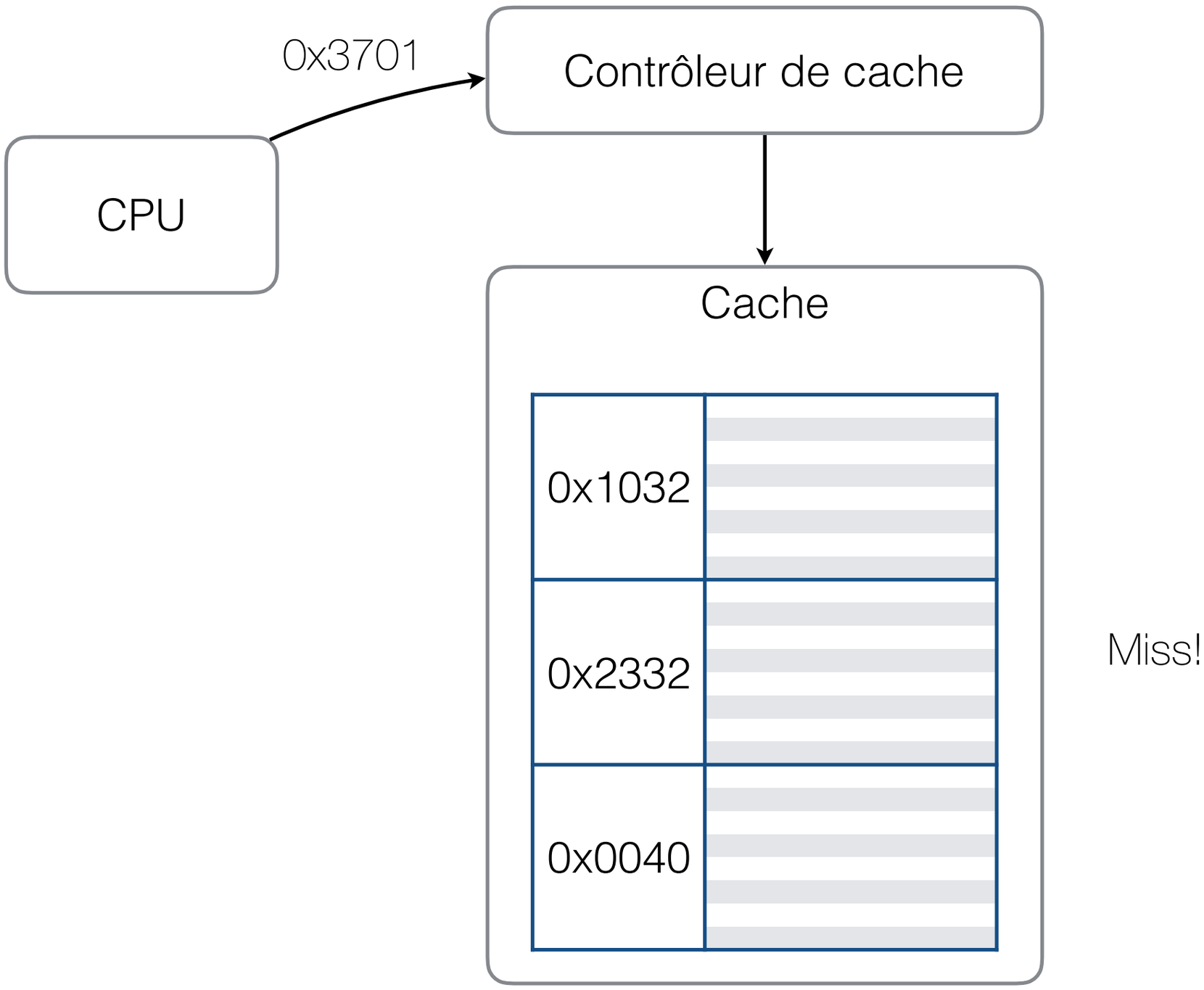
Écriture en cache « write-through »



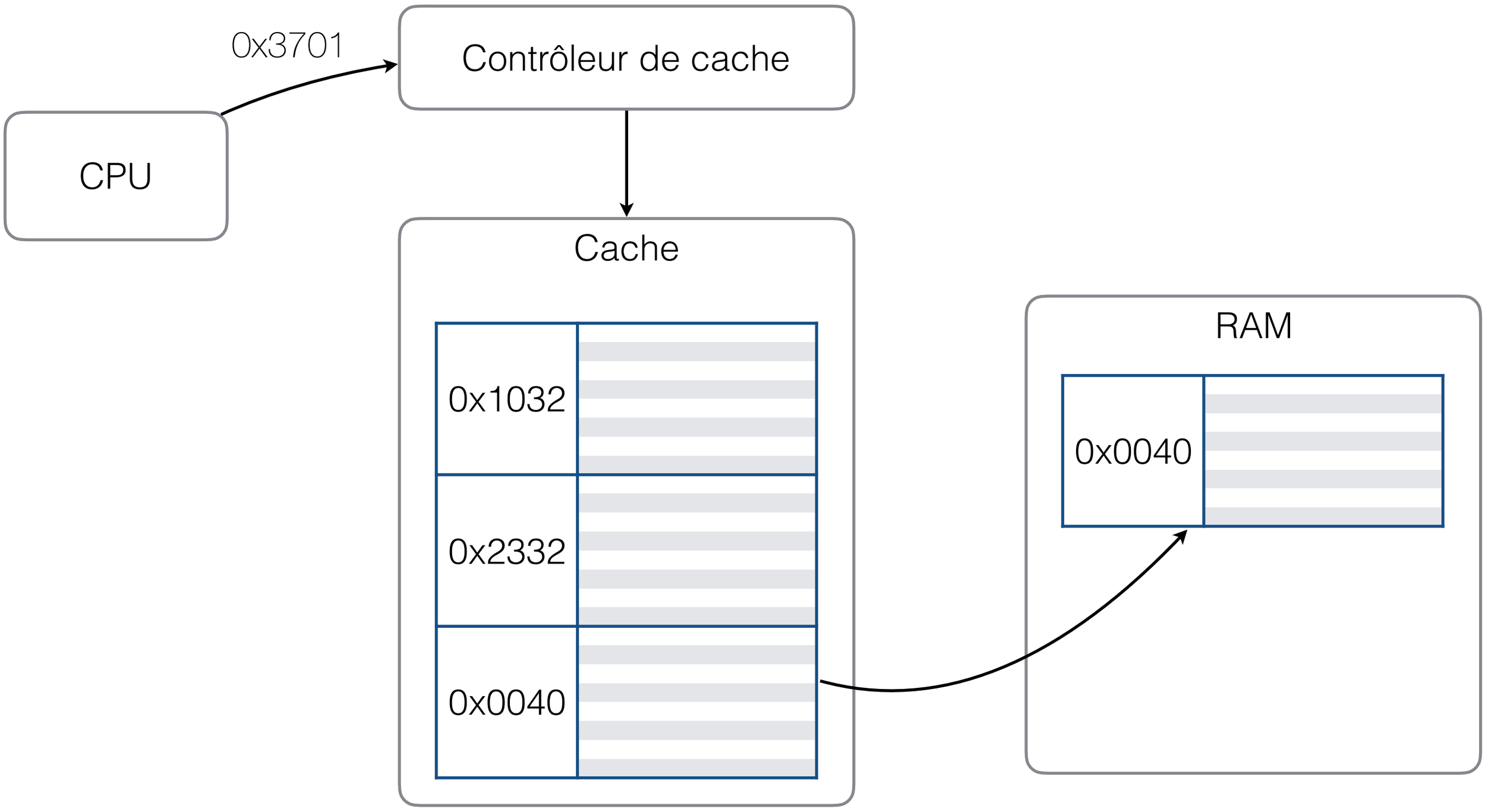
Écriture en cache « write-through »



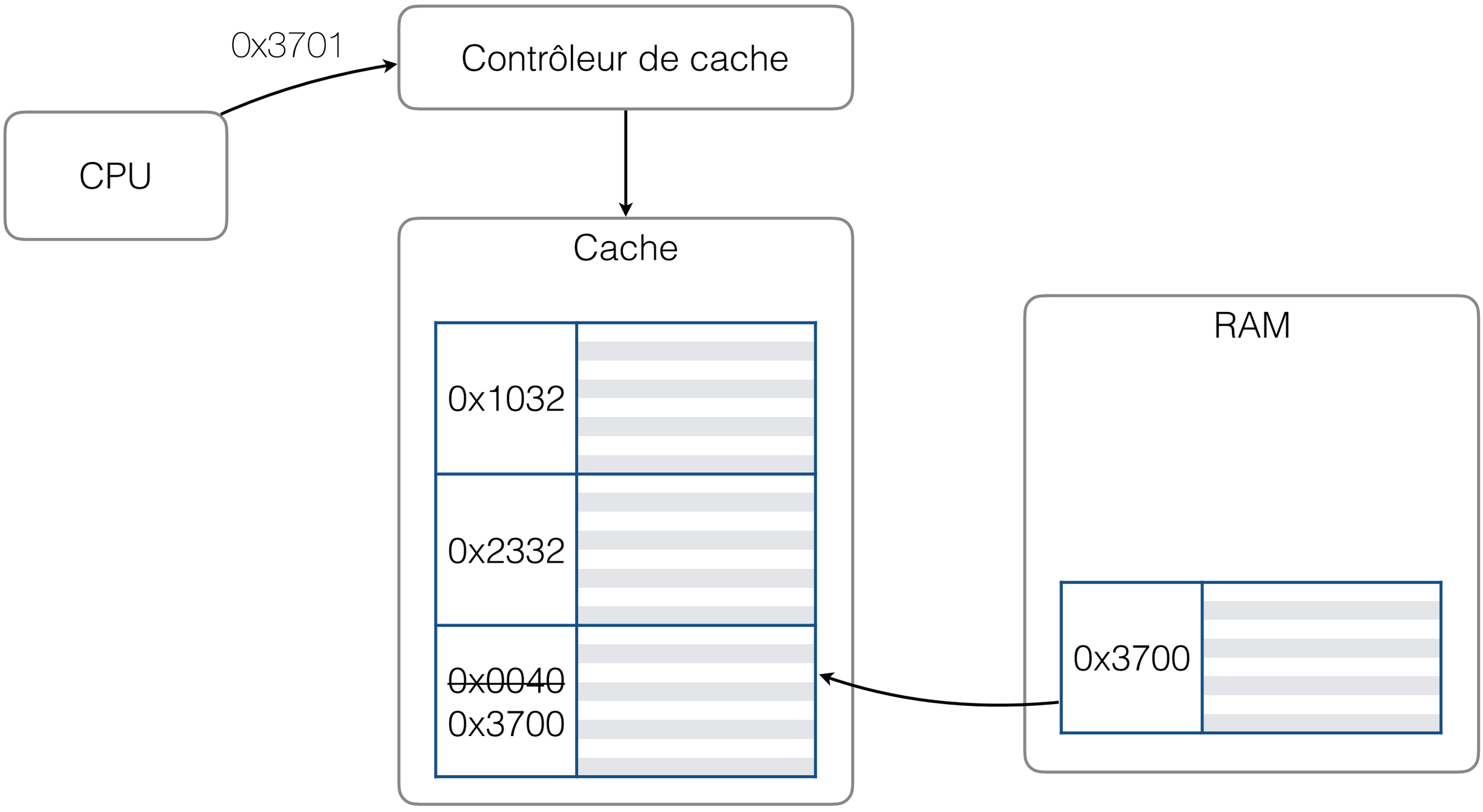
Écriture en cache « write-through »



Écriture en cache « write-through »

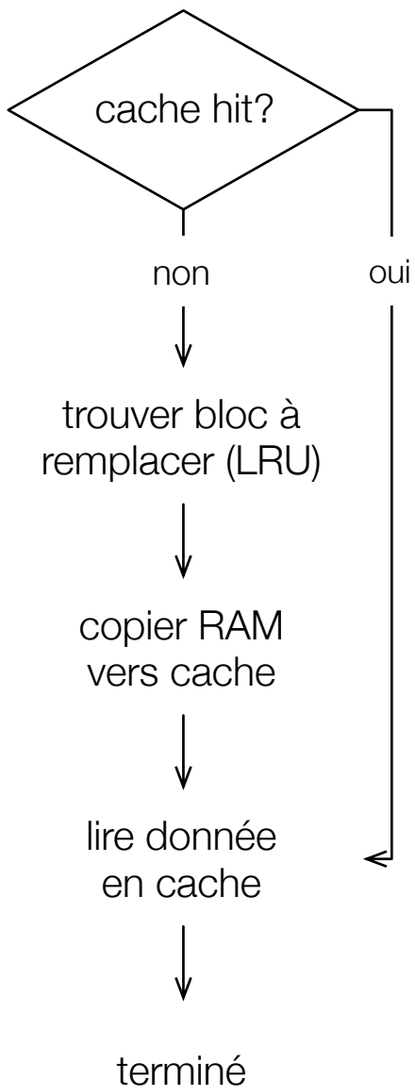


Écriture en cache « write-through »

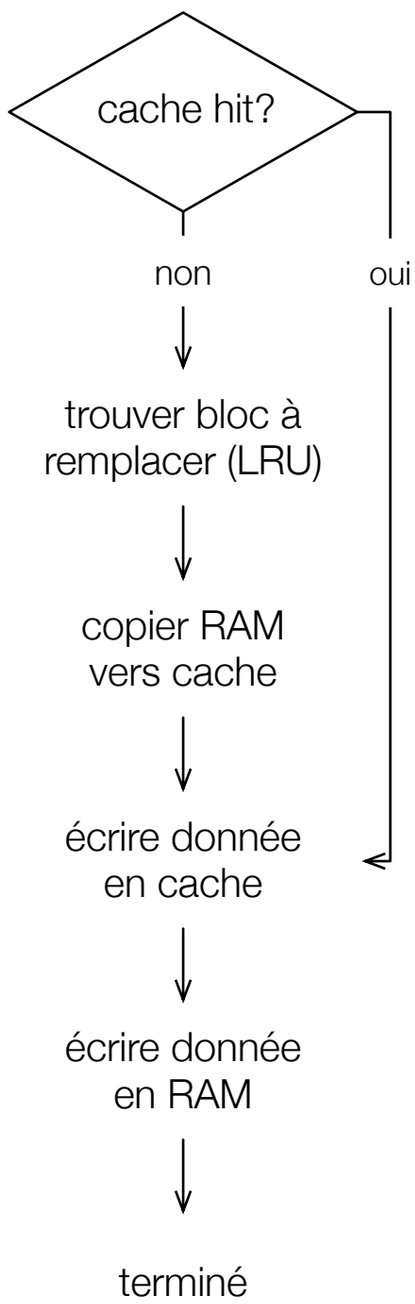


Cache « write-through »

Lecture



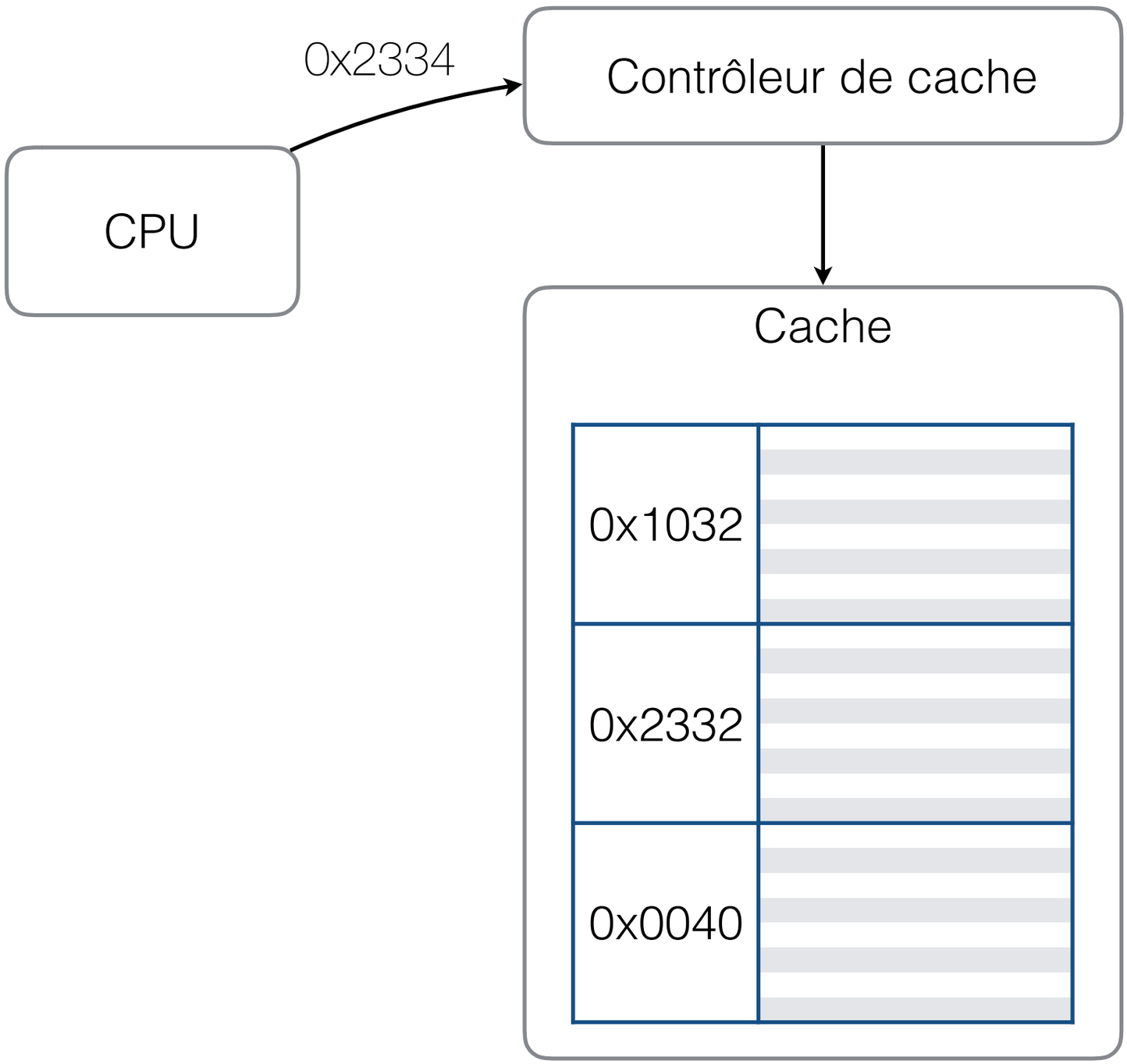
Écriture



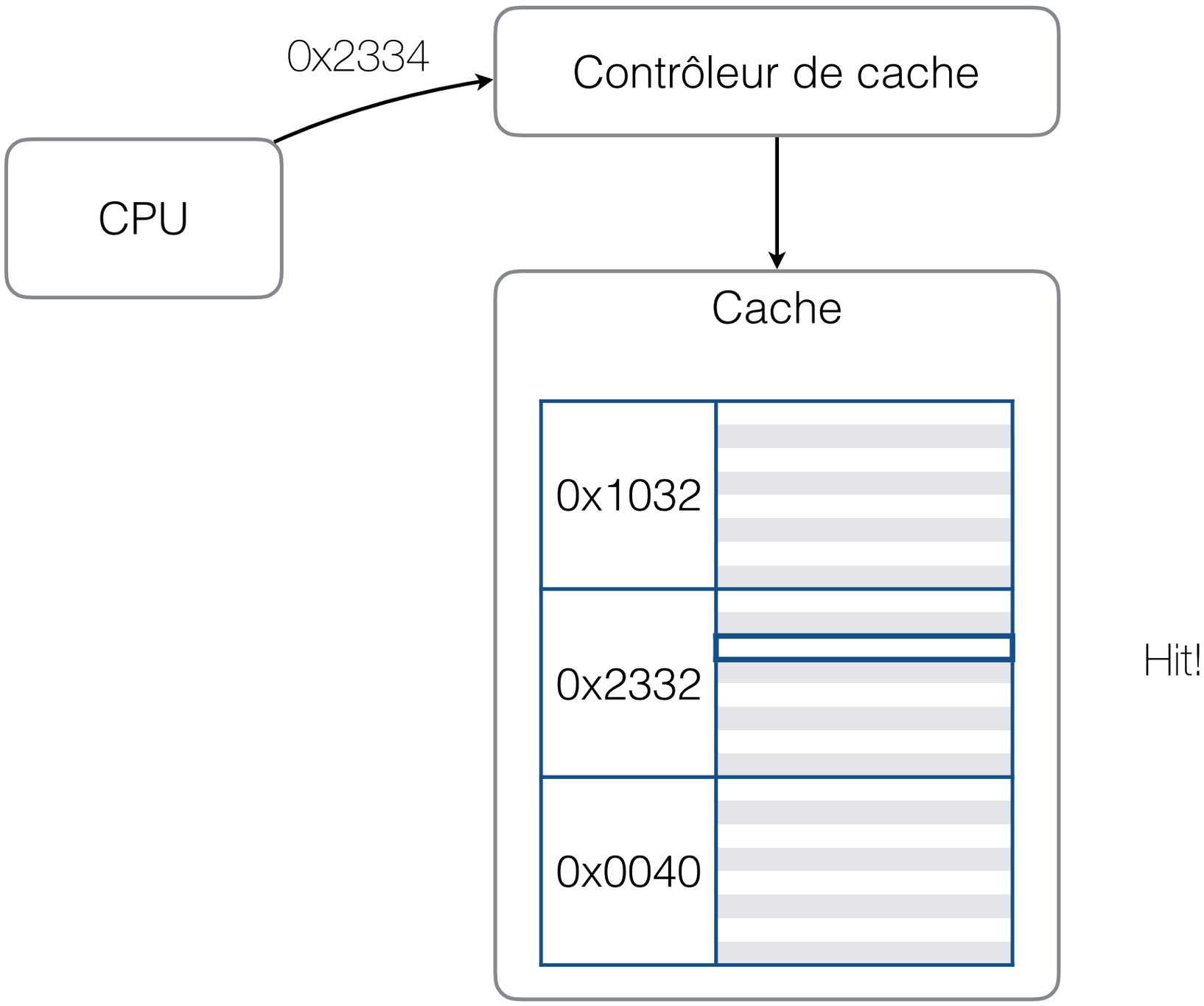
2 stratégies

- Que faire lorsque l'on veut écrire en cache?
  - « Write-through » : écrire les changements dans la RAM au fur et à mesure
    - L'exemple précédent utilisait cette technique
  - « **Write-back** » : écrire le bloc de données en RAM seulement lorsqu'il doit être remplacé
    - Il faut donc se rappeler que le bloc doit être remplacé

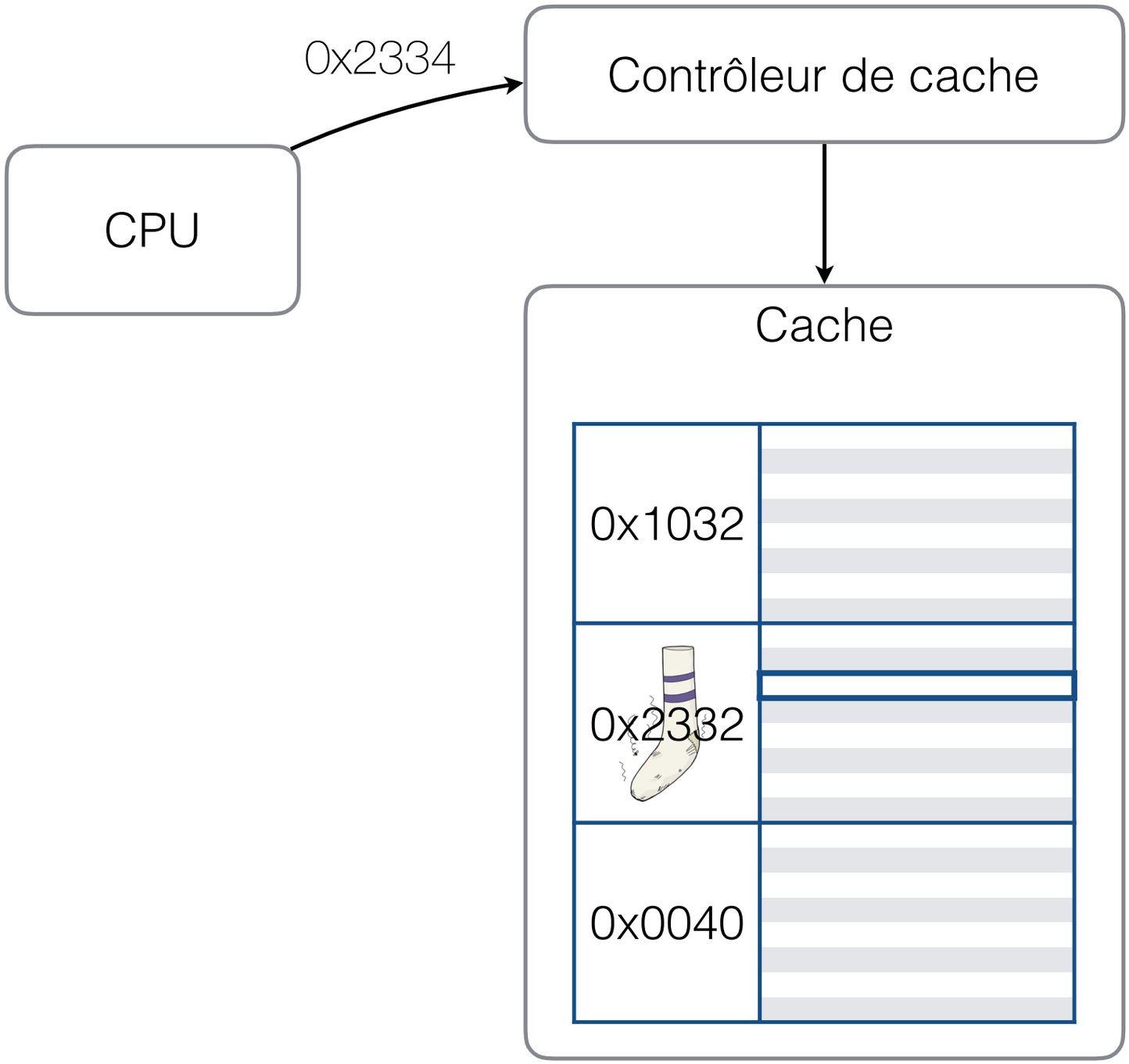
Écriture en cache « write-back »



Écriture en cache « write-back »

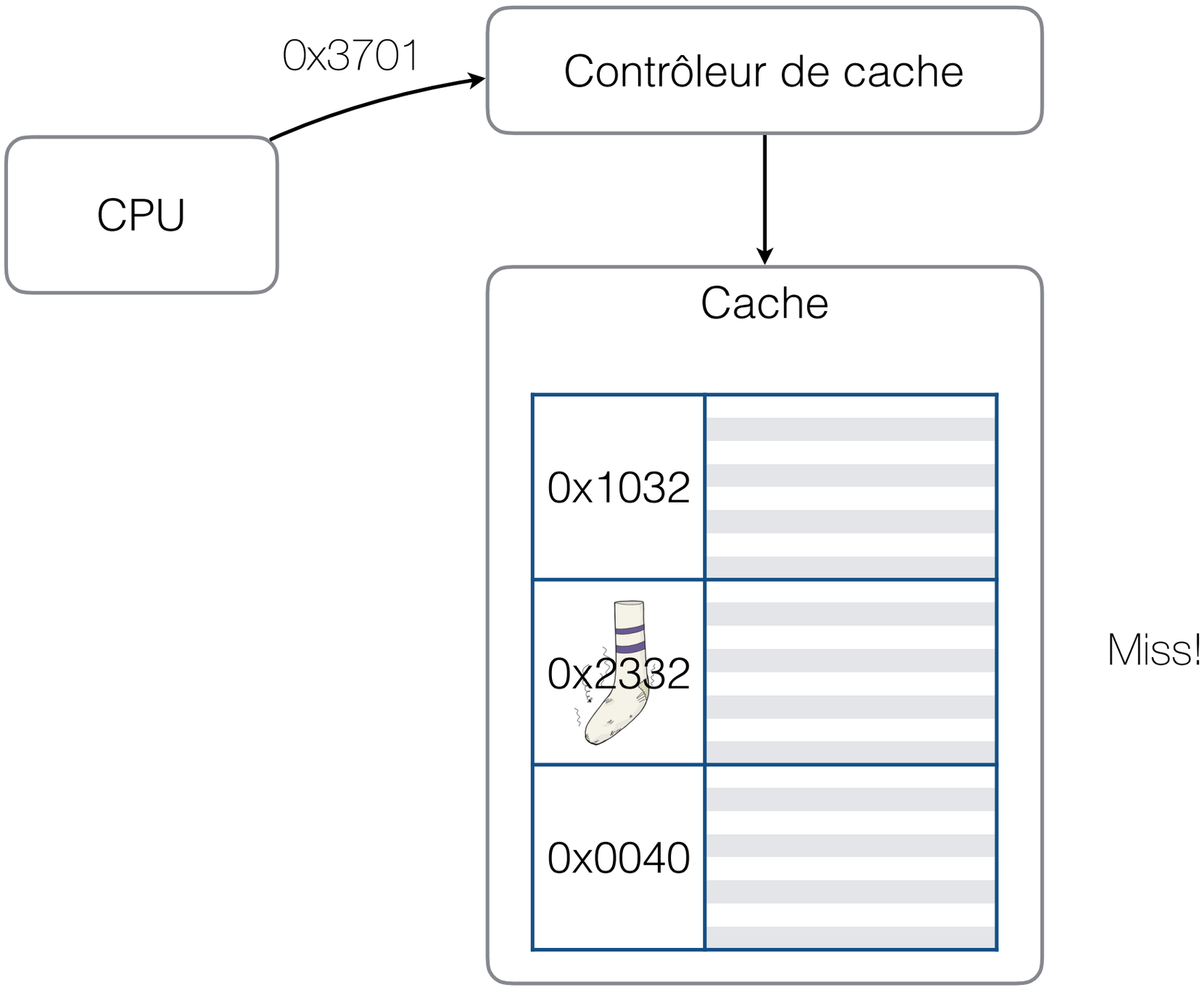


Écriture en cache « write-back »

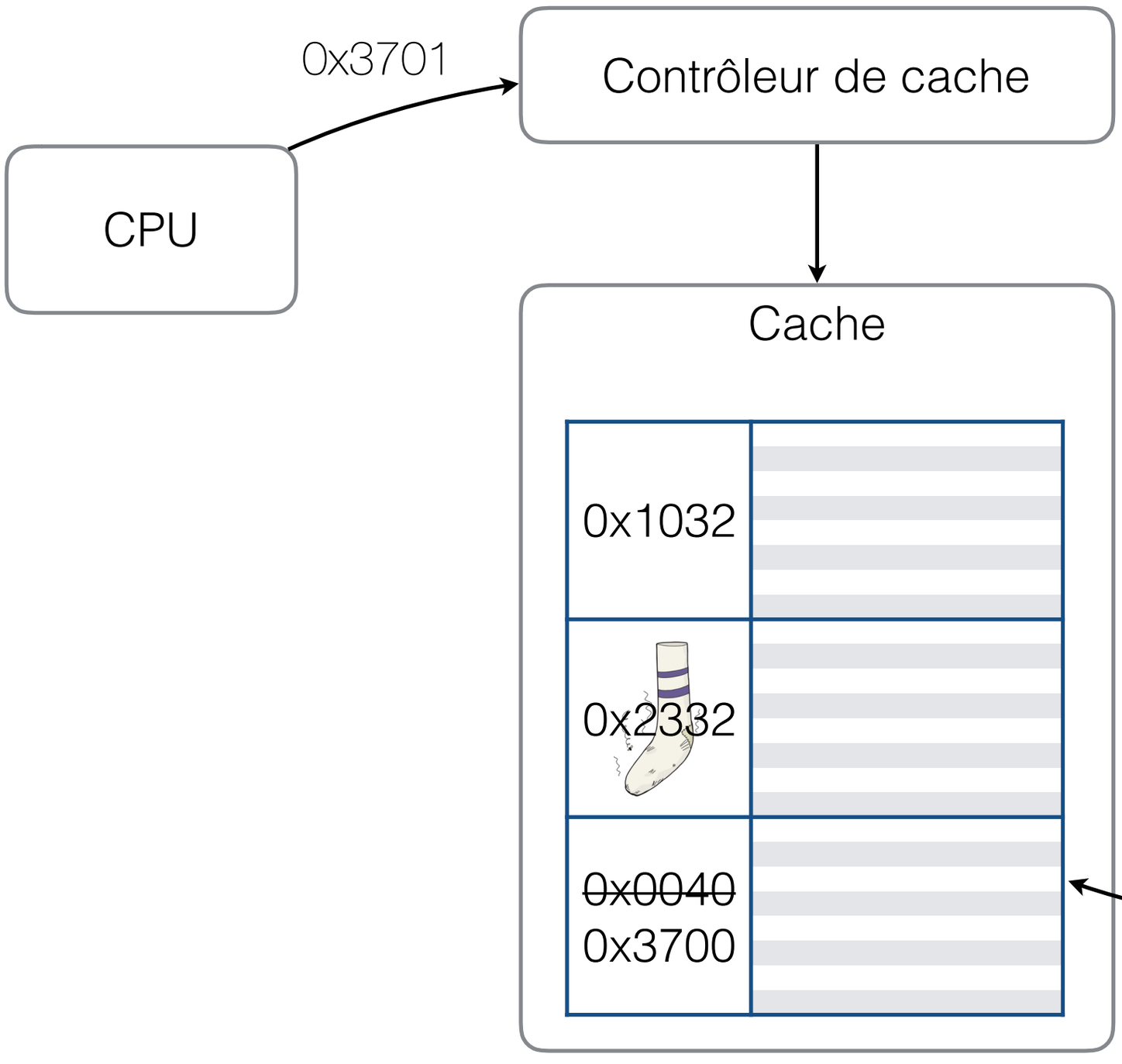


Le bloc est sale (« dirty »)!

Écriture en cache « write-back »

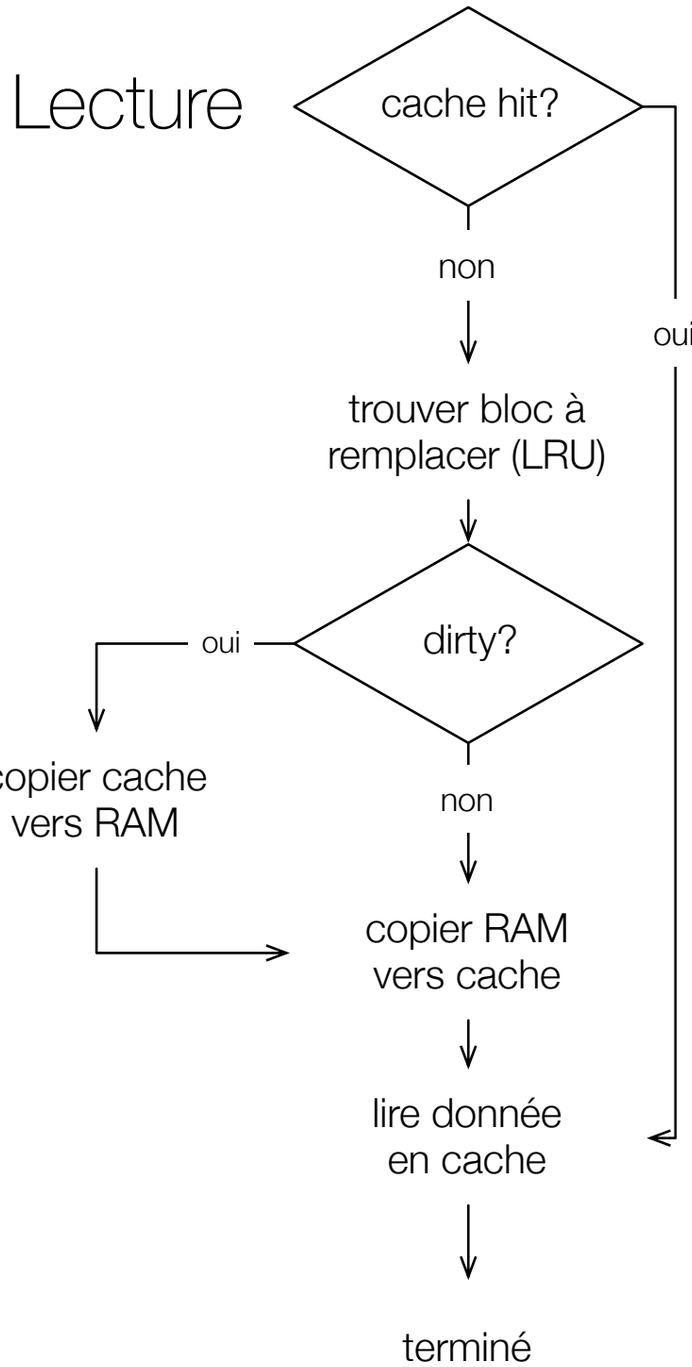


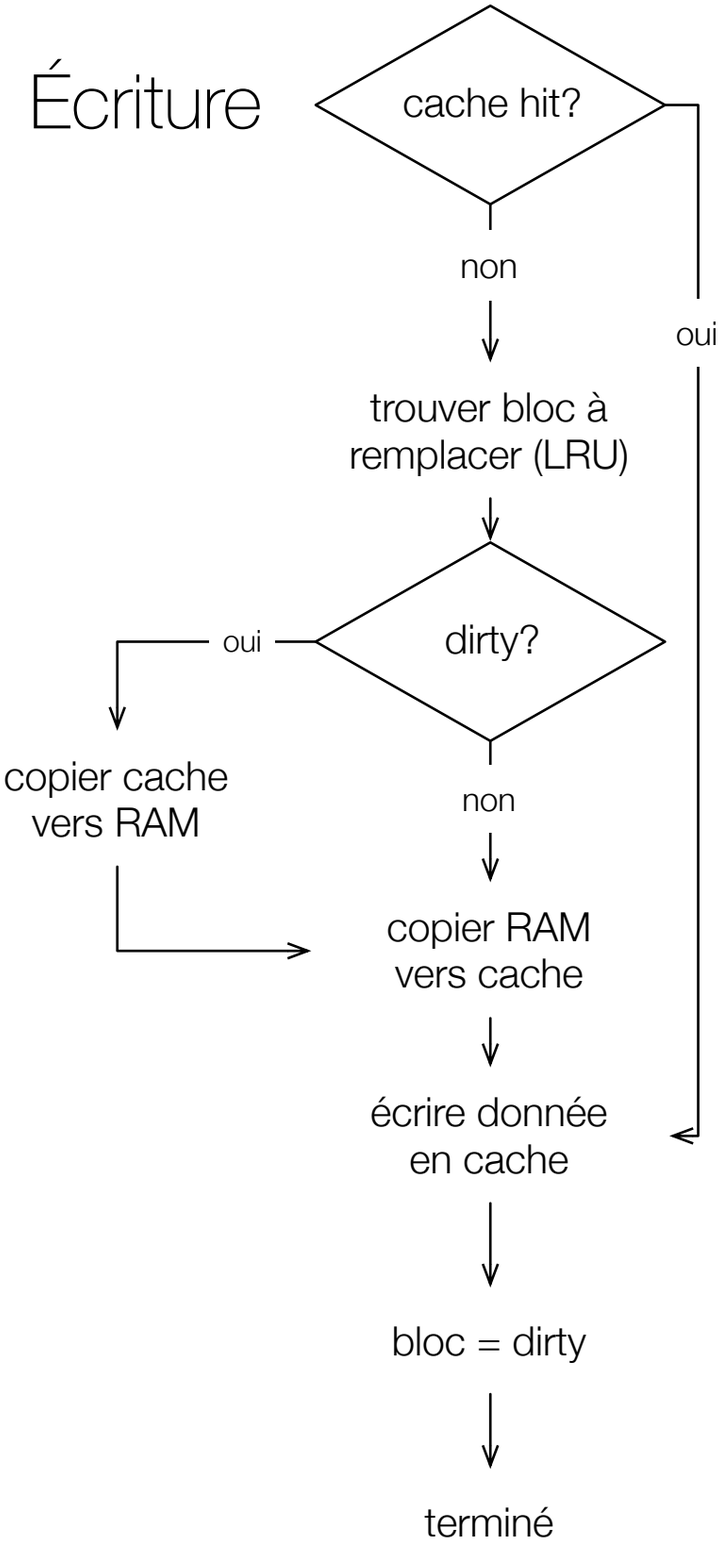
Écriture en cache « write-back »



Le bloc à remplacer est propre, alors pas besoin de l'écrire en RAM avant de le remplacer!

Cache « write-back »





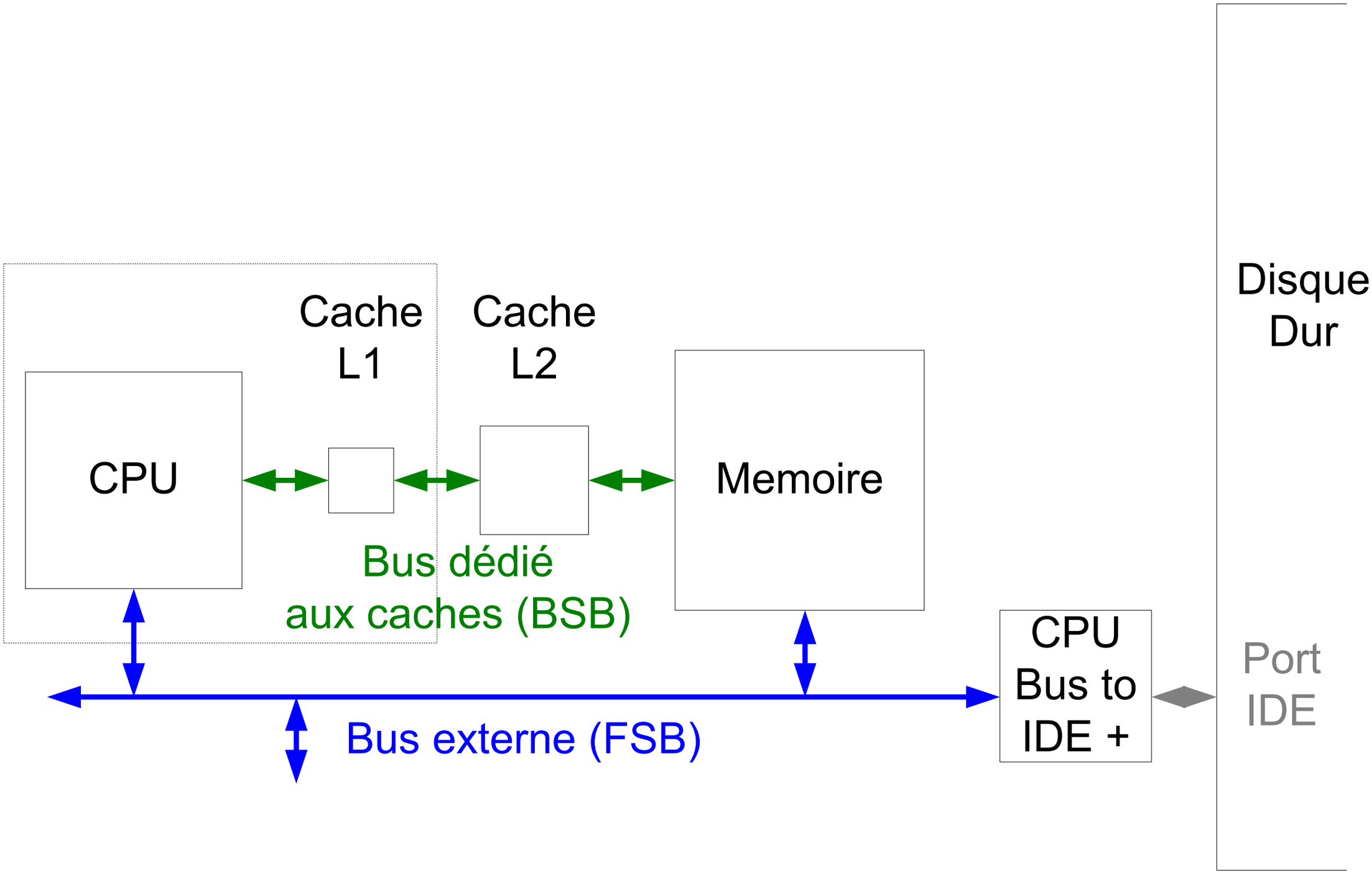
Récapitulation

Write-through

Mets la RAM à jour
à chaque écriture

Write-back

Hiérarchie de caches



Temps d’accès et vitesses de transfert

Type	Temps d’accès	Vitesse de transfert
Registres	0.25 ns (proc. 4 GHz)	très rapide!
Cache (SRAM)	1—10 ns	>> 1 GB/s
Mémoire vive (SDRAM)	10—20 ns	>> 1 GB/s
Flash (disques SSD)	25—100 µs en lecture 250 µs en écriture	200 MB/s—5 GB/s
Disque dur (magnétique)	3—15 ms	100 MB/s—1 GB/s
CD/DVD (optique)	100—500 ms	500 KB/s—4.5 MB/s
Backup (tape)	0.5 s et plus…	160 MB/s

Les caches

- Une cache est une mémoire d'accès très rapide placée en tampon entre une unité rapide et une mémoire lente afin d'accélérer les accès mémoire.
- La taille de la cache est déterminée par la différence de vitesse entre l'unité rapide (exemple: le microprocesseur) et la mémoire lente
- La cache contient les données de la mémoire lente les plus susceptibles d'être demandées par l'unité rapide. En raison du principe de localité, ce sont les dernières données/ instructions utilisées par le microprocesseur et celles adjacentes.
- Les caches contiennent des blocs de données plutôt que des données individuelles. Selon le principe de localité, il y a de fortes chances que deux instructions qui se suivent soient dans le même bloc.
- Lorsqu'il veut une donnée, le microprocesseur cherche d'abord dans les caches. Un hit survient lorsque la donnée est trouvée.
- Si le microprocesseur ne trouve pas les données dans les caches (miss), il cherche dans la mémoire. S'il trouve, les données sont copiées dans les caches, puis transférées au microprocesseur. Les données adjacentes sont également transférées dans la cache.
- Le hit ratio est la probabilité de retrouver une donnée dans la mémoire cache.

Les caches

- Il y a plusieurs façons de relier les blocs de mémoires aux blocs de cache. Vous retrouverez l'association directe qui consiste à relier un nombre fixe de blocs de mémoires à chaque emplacement de cache (méthode simple, peu coûteuse mais pas optimale). Il y a aussi les caches associatives dans lesquelles chaque bloc de mémoire peut se retrouver n'importe où dans la cache (complexe, coûteux, optimal). Finalement, il y a des caches partiellement associatives (associatives par set) qui sont un compromis entre les deux méthodes de mapping: chaque bloc de mémoire peut se retrouver dans un nombre fini de blocs de cache (bon compromis!).

- Lorsque le microprocesseur écrit une variable en mémoire, il doit également tenir compte des caches. Il existe deux modes d'écriture principaux pour les caches et la mémoire: write-through et write-back. En write-through le microprocesseur écrit une donnée simultanément dans la mémoire et dans la cache ce qui ne minimise pas les accès mémoire lents. En write-back, le microprocesseur écrit dans la cache seulement. Les données sont transférées de la cache à la mémoire lorsque le bloc de cache doit être remplacé. Cela semble optimal, mais le DMA devient complexe (voir plus loin).

- Lorsque les caches sont pleines, il faut remplacer une donnée. Il existe plusieurs algorithmes de remplacement qui dépendent de la façon dont les blocs de caches et les blocs de mémoire sont reliés.

- Il y a parfois 1 cache, parfois 2 ou même 3. Avoir plusieurs niveaux de caches assure un bon hit ratio.

La cache L1

- La cache L1 est habituellement à l'intérieur du même circuit intégré que le microprocesseur, souvent imbriquée dans l'architecture même du processeur.
- Petite, car espace sur le microprocesseur limité (128 ko pour les premiers ATHLON, 32 ko pour les pentiums 2/3)
- Divisée en deux pour les données et pour les programmes.
- Très utilisée.

Les caches L2, L3 et autres

- Les ordinateurs modernes ont au moins deux niveaux de cache et souvent trois.
- La cache L2 est plus grosse que la cache L1 (256ko à 2Mo).
- Les caches L2 et L3 sont habituellement à l'extérieur du microprocesseur (mais de plus en plus à l'intérieur). Indépendantes de l'architecture du microprocesseur lorsqu'à l'extérieur.
- Habituellement, les données et les instructions sont gérées de la même façon dans les caches L2 et L3 (elles sont dites unifiées). Cependant, les instructions et les données sont de plus en plus souvent séparées dans les caches de niveau 2 (L2) voire de niveau 3.
- Moins rapide que L1 mais environ 10 fois plus rapide que la mémoire.
- Dans certains systèmes, il y a 4 niveaux de cache…
- Dans les ordinateurs modernes, il y a des caches dans les disques durs, et pour plusieurs périphériques. Ces caches contiennent les données du disque ou celles du périphérique qui les sont plus susceptibles d'être accédées prochainement.

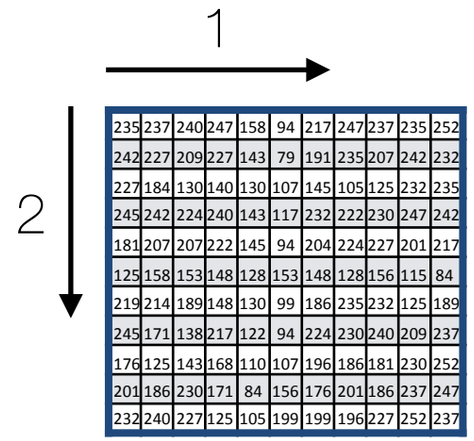
Exercice

- Décrivez les étapes nécessaires pour que le micro-processeur écrive une donnée en mémoire si:
  - l’adresse mémoire n’est dans aucune cache;
  - il y a deux niveaux de cache (L1 et L2);
  - le système utilise la stratégie “write-through”.

Retour

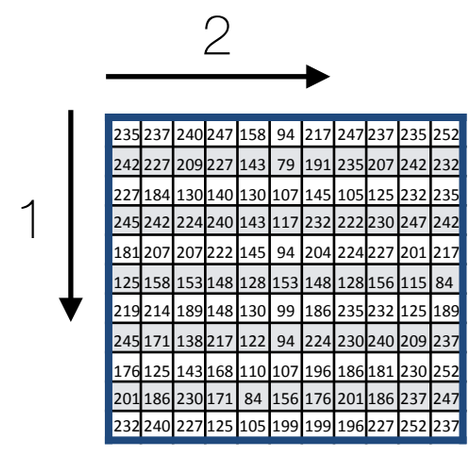
- Laquelle de ces deux versions est la plus rapide?

```
for (int i = 0; i < 256; i++) {
  for (int j = 0; j < 512; j++) {
    img[i*512 + j] = 0;
   }
 }
```



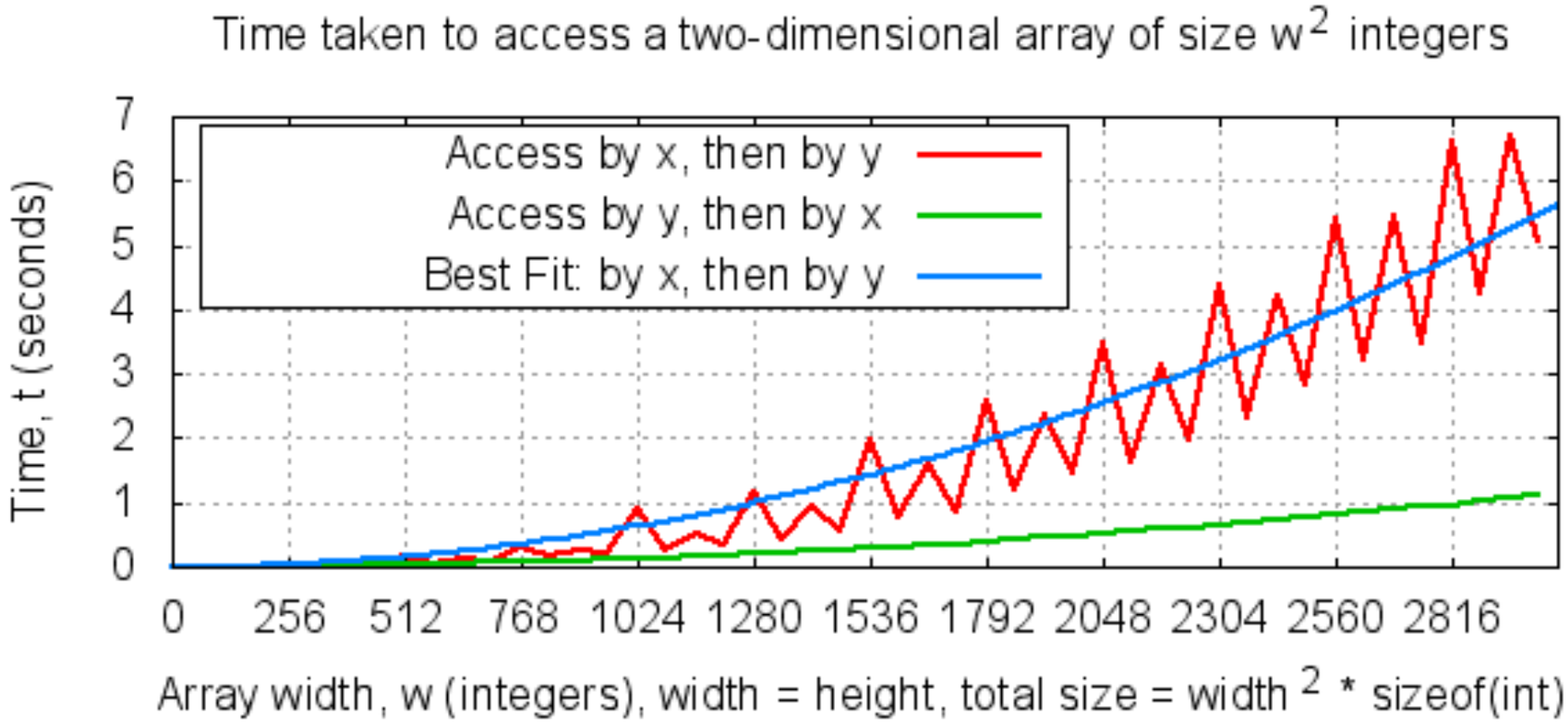
ou

```
for (int j = 0; j < 512; i++) {
  for (int i = 0; i < 256; j++) {
    img[i*512 + j] = 0;
   }
 }
```



i = lignes
j = colonnes

Ordre d'accès mémoire



crédit: http://www.tophatstuff.co.uk/archive.php?p=119

La mémoire principale, une cache du disque dur?

- La mémoire contient des données et des instructions regroupées en pages (souvent 4K).
- La mémoire fournit des informations aux caches.
- Si les données ne sont pas trouvées dans la mémoire (page fault), il faut chercher dans le disque dur (très long). La page contenant la donnée recherchée est habituellement transférée en mémoire. Elle remplace souvent une page déjà en place qu'il faut sauvegarder d'abord (swap page).

F.A.Q.

- Q. Où est la table des pages?
  - A. En mémoire
- Q. Combien d'accès mémoire doit-on faire pour accéder à des données?
  - A. 2: un pour accéder à la table des pages, un pour accéder aux données
- Q. Que faire pour éviter les accès mémoires doublés?
  - A. On utilise une "cache" spécialisée: le TLB!

Traduction d'adresse, MMU et TLB

- Le MMU d'une mémoire paginée est plus complexe que celui d'une mémoire allouée de façon contiguë. En effet, pour tous les accès à la mémoire, il faut accéder à la table de pages afin de décoder l'adresse à lire. Comme la table de page est souvent volumineuse, elle est elle-même en mémoire: il faut faire deux lectures de la mémoire pour aller chercher une instruction, ce qui est très long et quasi-inadmissible.

- Le TLB ("Translation Lookaside Buffer"), est une table de registres très rapides à l'intérieur du CPU. Le TLB contient quelques entrées de la table de pages, les dernières utilisées. Il s'agit d'une forme de cache. Pour faire de la traduction d'adresse, le MMU regarde d'abord si la page à traduire est dans le TLB. Puisque le TLB contient les dernières entrées de la page de table utilisées, alors le MMU y trouve presque toujours la page à lire. Lorsqu'un hit se produit dans le TLB, le temps d'accès à la mémoire n'est pas augmenté par la pagination.

- Un TLB typique contient 8 à 4096 entrées de page de table. Lorsque la page est trouvée dans le TLB, ½ cycle à 1 cycle d'horloge est requis pour faire la translation d'adresse. Lorsque la page n'est pas dans le TLB, il faut faire deux accès mémoire… Le taux de succès dans le TLB est entre 99% et 99.99%!

Mémoire virtuelle vs cache

- Cache
  - accélérer la vitesse d'accès mémoire
- Mémoire virtuelle
  - augmente la "quantité perçue" de mémoire disponible
  - indépendant de l'architecture

Exercice 2: accès mémoire paginée avec caches

- Un système possède les caractéristiques suivantes:
  - la mémoire est paginée;
  - il y a deux niveaux de cache (L1 et L2) utilisant la stratégie « write-back »
- Décrivez les étapes nécessaires pour qu'un programme puisse lire une donnée à une adresse virtuelle si:
  - l'entrée dans la table des pages est disponible dans le TLB;
  - il y a faute de page;
  - l'adresse mémoire (physique) n'est dans aucune cache;
  - le bloc le moins récemment utilisé (LRU) n'est pas "dirty".

Références et exercices

- Références
  - Irv Englander: chapitre 7 (jusqu'à 7.6), sections 8.3 (caches seulement), chapitre 9