La mémoire cache

GIF-1001 Ordinateurs: Structure et Applications

Jean-François Lalonde

Image: une matrice de pixels

Image: une matrice de pixels

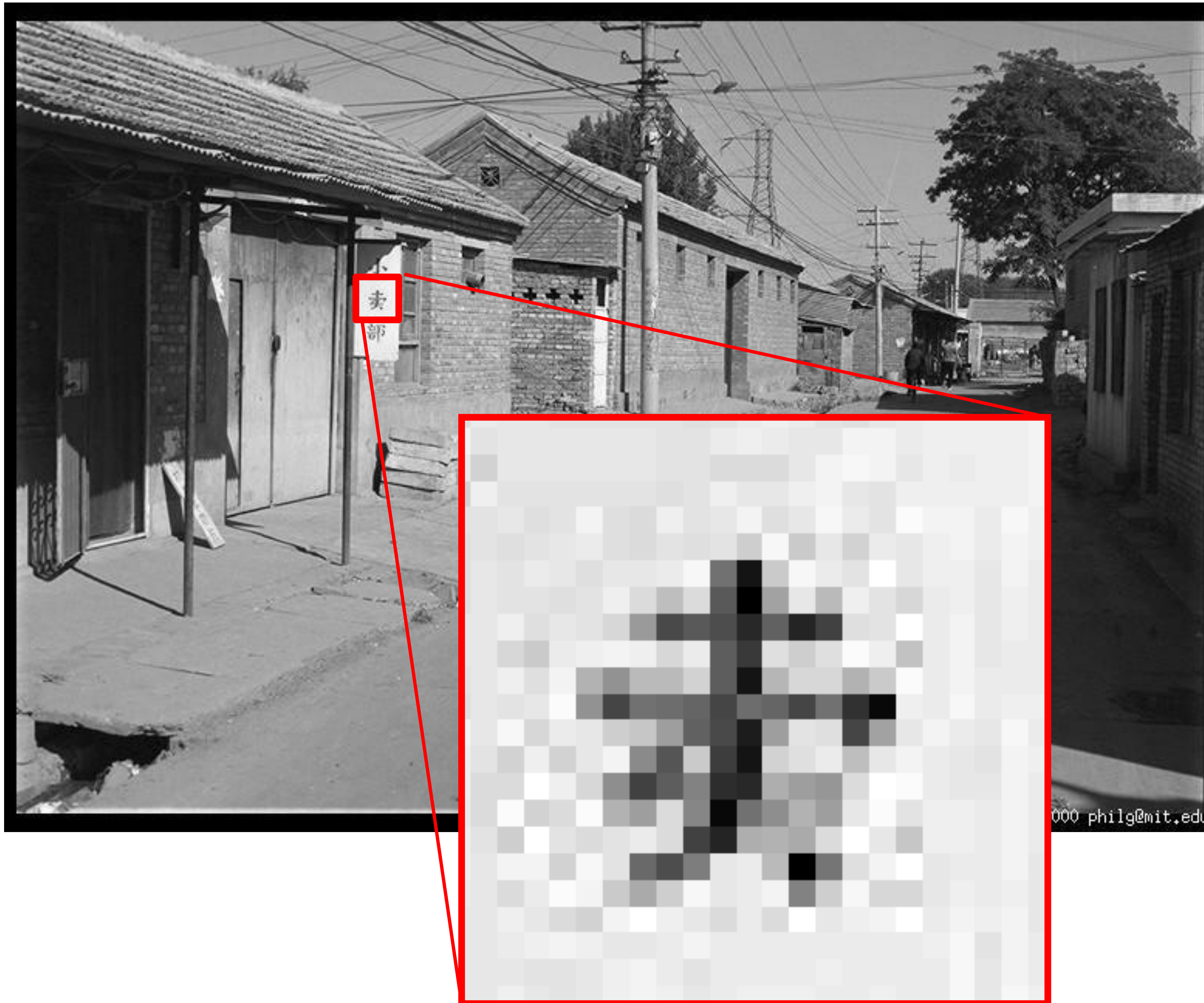
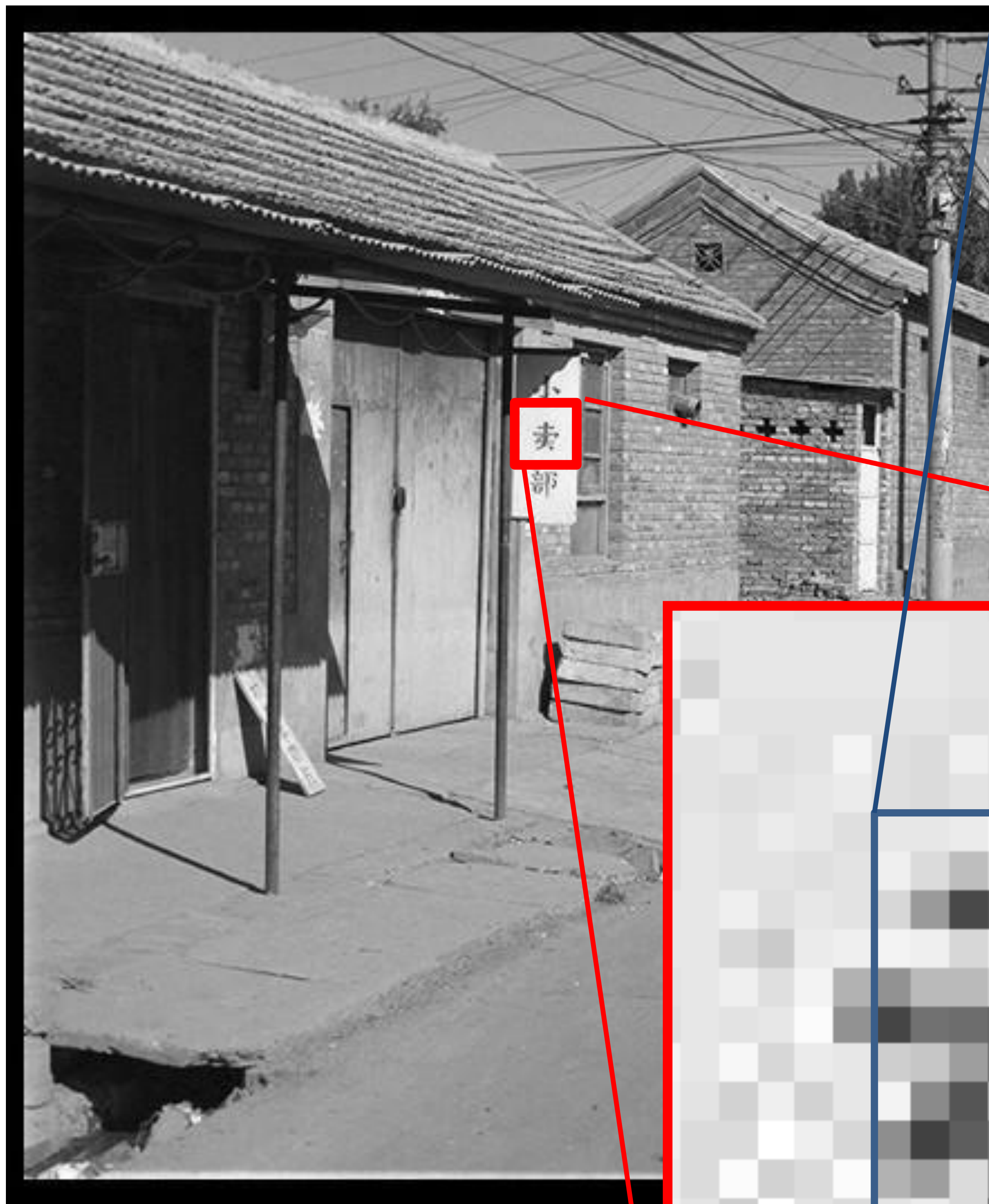


Image: une matrice de pixels



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

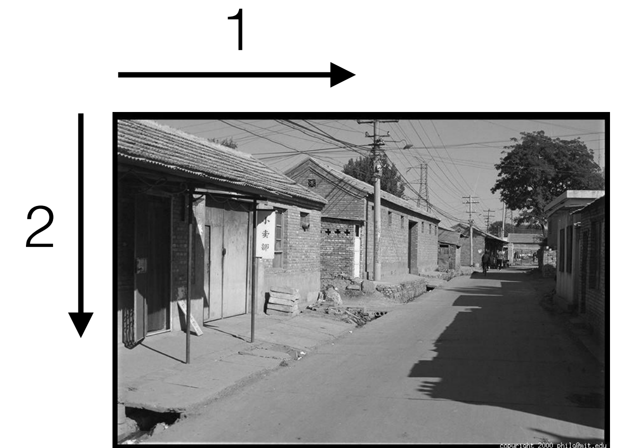
Parcourir une image

Question
Est-ce qu'une de ces deux options est plus rapide qu'une autre?
Si oui, laquelle?

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

1

2

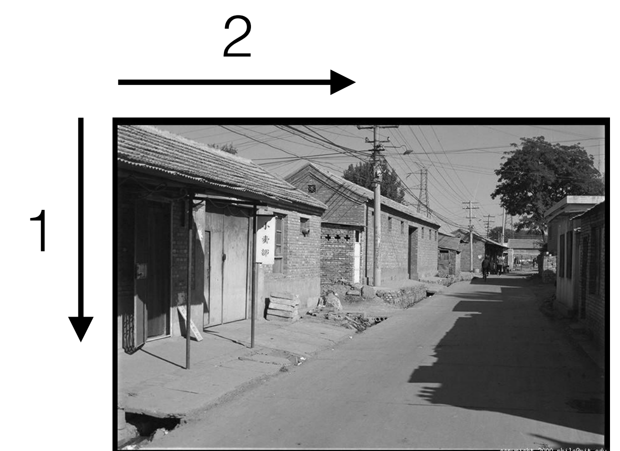


ou

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

2

1

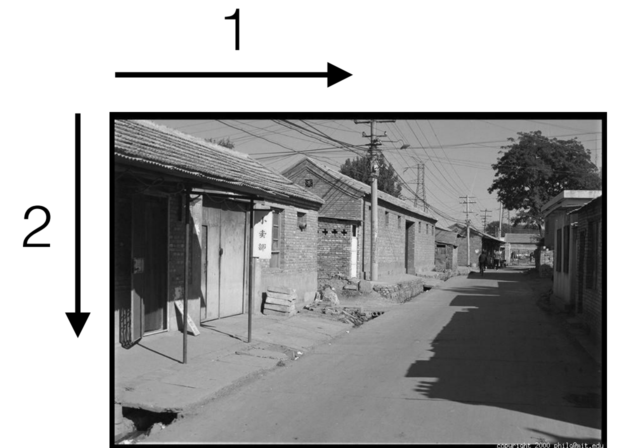


Ici, `i` désigne les lignes ("y"), et `j` les colonnes ("x").

Parcourir une image

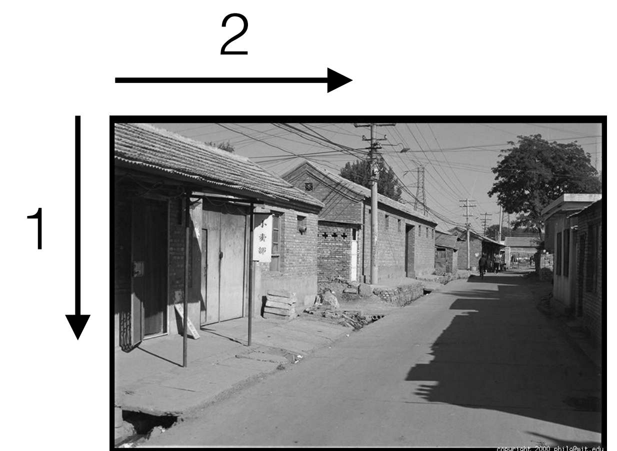
Réponse
La première version est la plus rapide!
Mais **pourquoi**?

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



ou

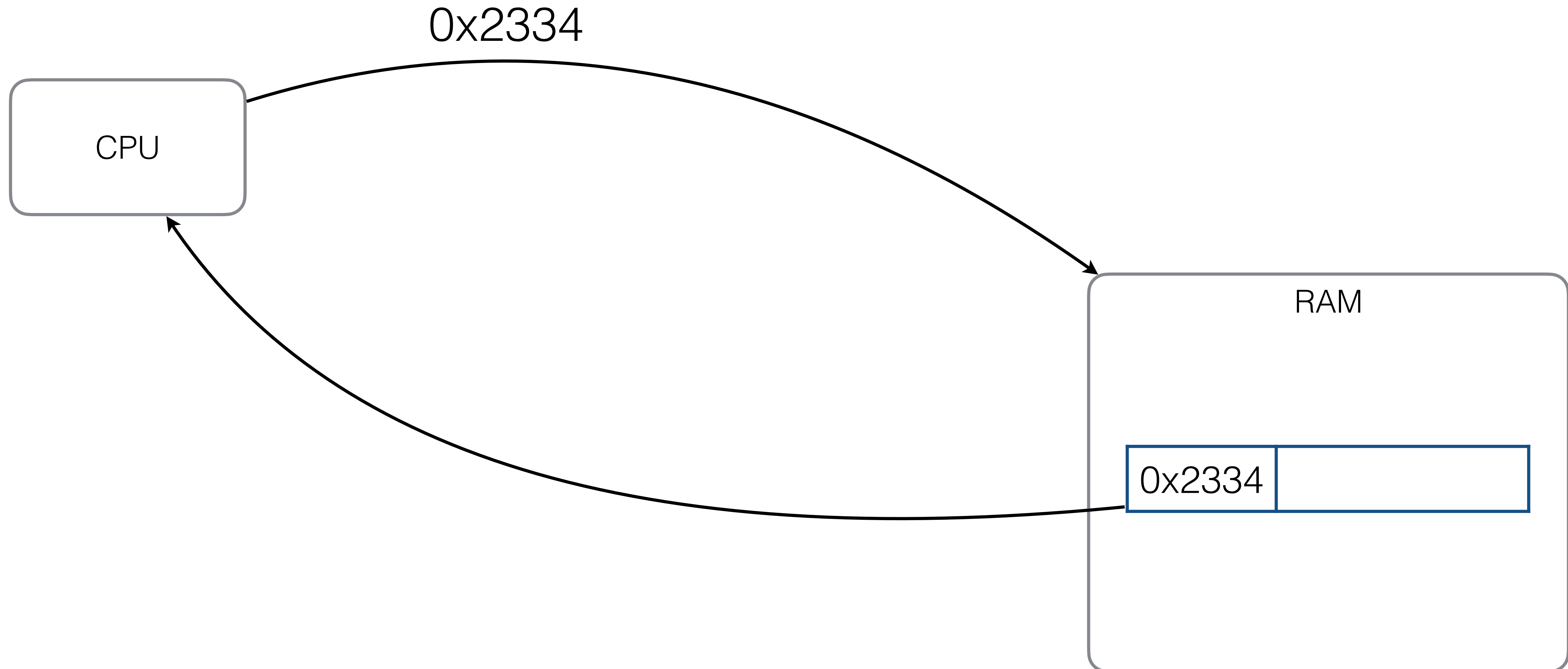
```
for (int j = 0; j < 512; i++) {
  for (int i = 0; i < 256; j++) {
    img[i*512 + j] = 0;
   }
 }
```



Ici, `i` désigne les lignes (“y”), et `j` les colonnes (“x”).

Lecture en mémoire

Comment faire pour **accélérer** les accès mémoire?



On rajoute une cache!

- Cache: mémoire **très rapide** placée entre:
  - le microprocesseur (rapide!) et;
  - une mémoire (moins rapide)
- La cache est présente pour **accélérer les accès mémoire**.

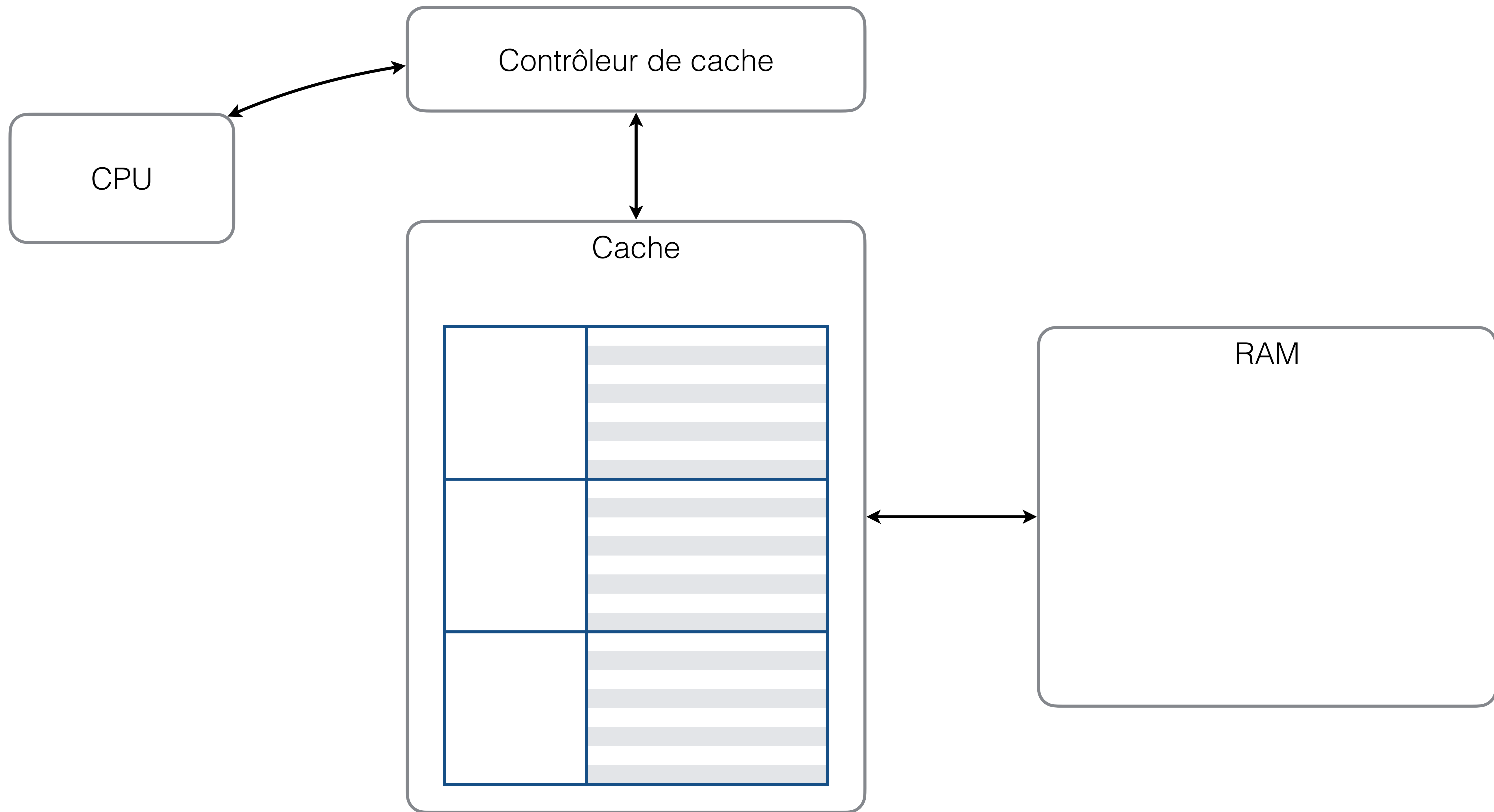
Type	Temps d'accès	Vitesse de transfert
Registres	0.25 ns (proc. 4 GHz)	très rapide!
Cache (SRAM)	1—10 ns	>> 1 GB/s
Mémoire vive (SDRAM)	10—20 ns	>> 1 GB/s

On rajoute une cache!

- La taille d’une cache est **plus petite** que la mémoire à laquelle elle est branchée

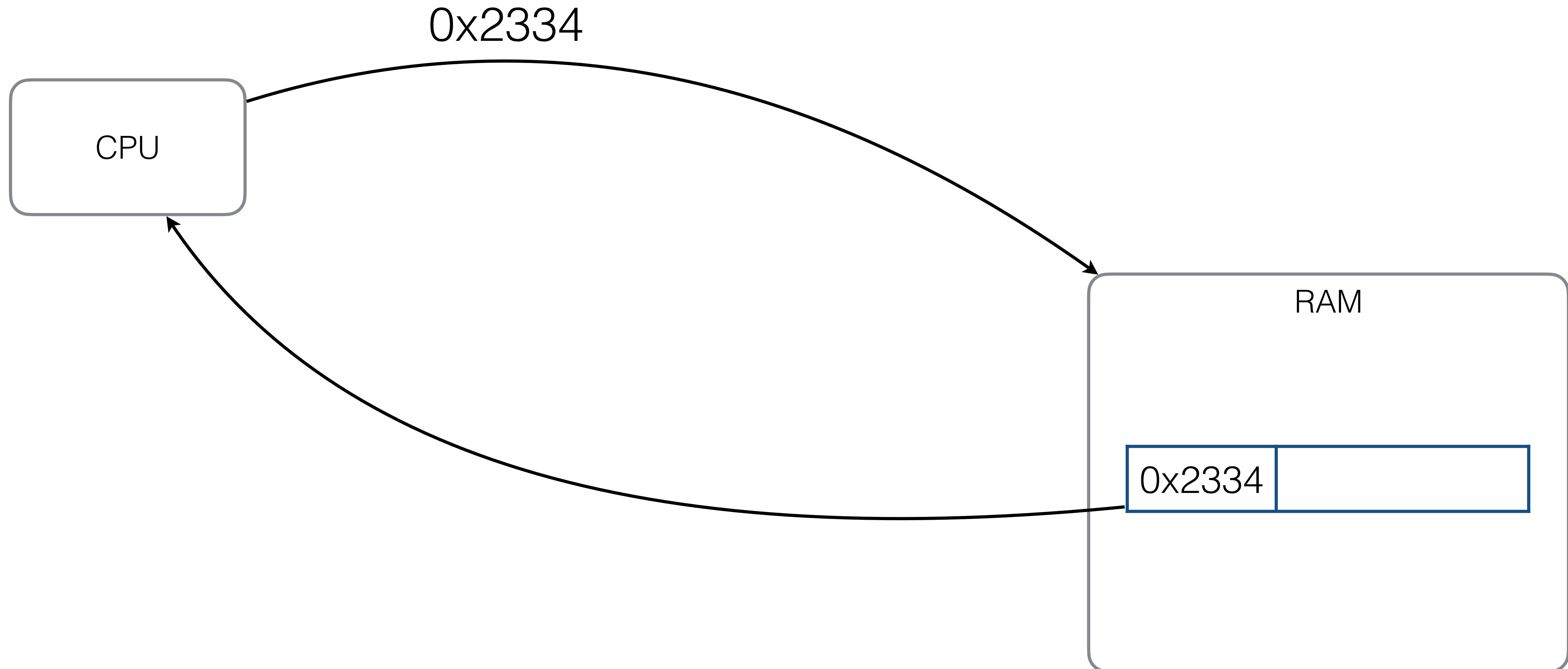
- La cache contient des **blocs** de données plutôt que des données individuelles.

On rajoute une cache!



Lecture en mémoire

Comment faire pour accélérer les accès mémoire?

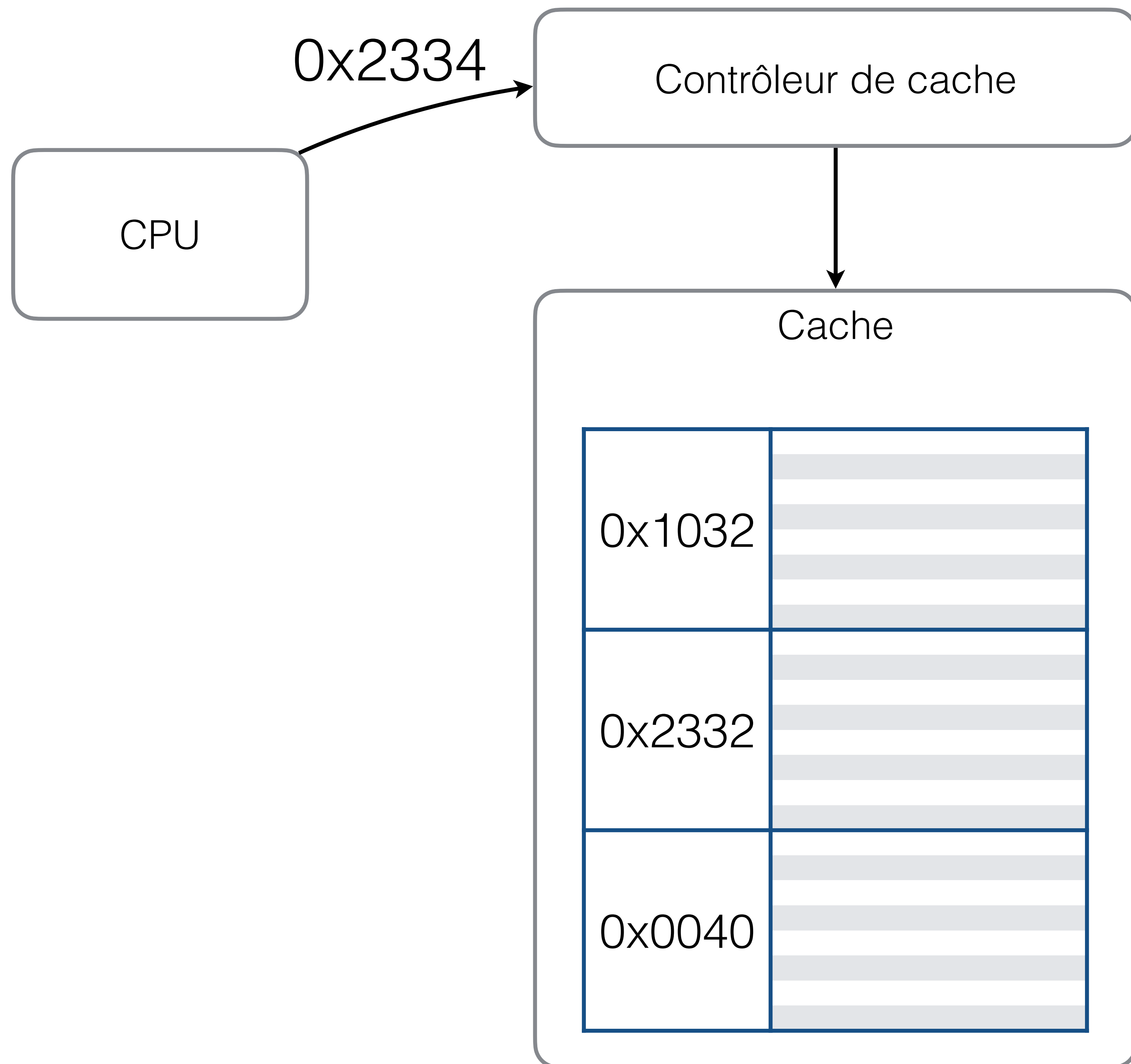


On rajoute une cache!

Lorsque le microprocesseur fait un accès mémoire, on passe **toujours** par la cache en premier.

Il y a 2 scénarios possibles.

L'adresse peut être **présente** ou **absente** en cache.



RAM

Lecture en cache

0x2334

Contrôleur de cache

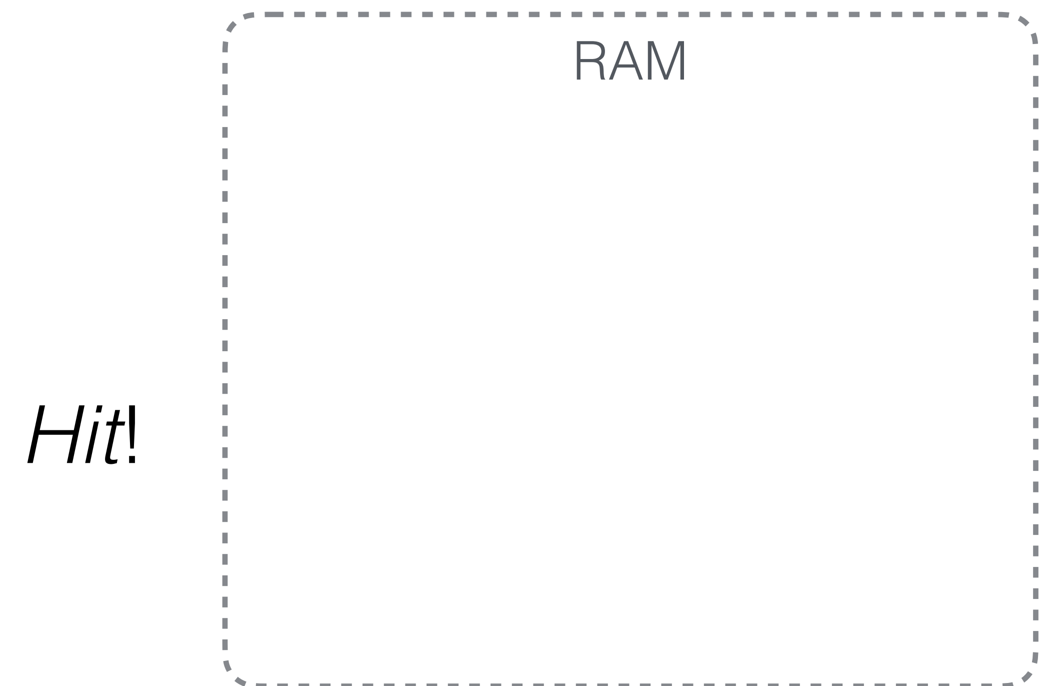
CPU

Cache

0x1032	
0x2332	
0x0040	

Si l’adresse est **présente** en cache,
on dit qu’il y a un *hit*.

Hit!



Lecture en cache

0x3701

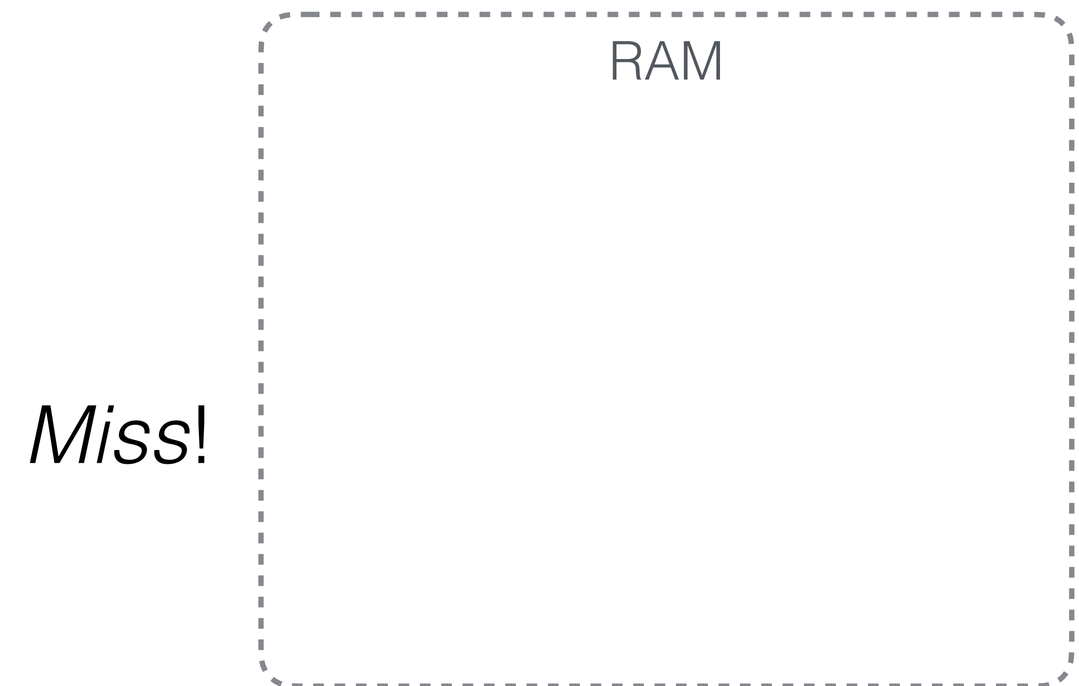
Contrôleur de cache

CPU

Cache

0x1032	
0x2332	
0x0040	

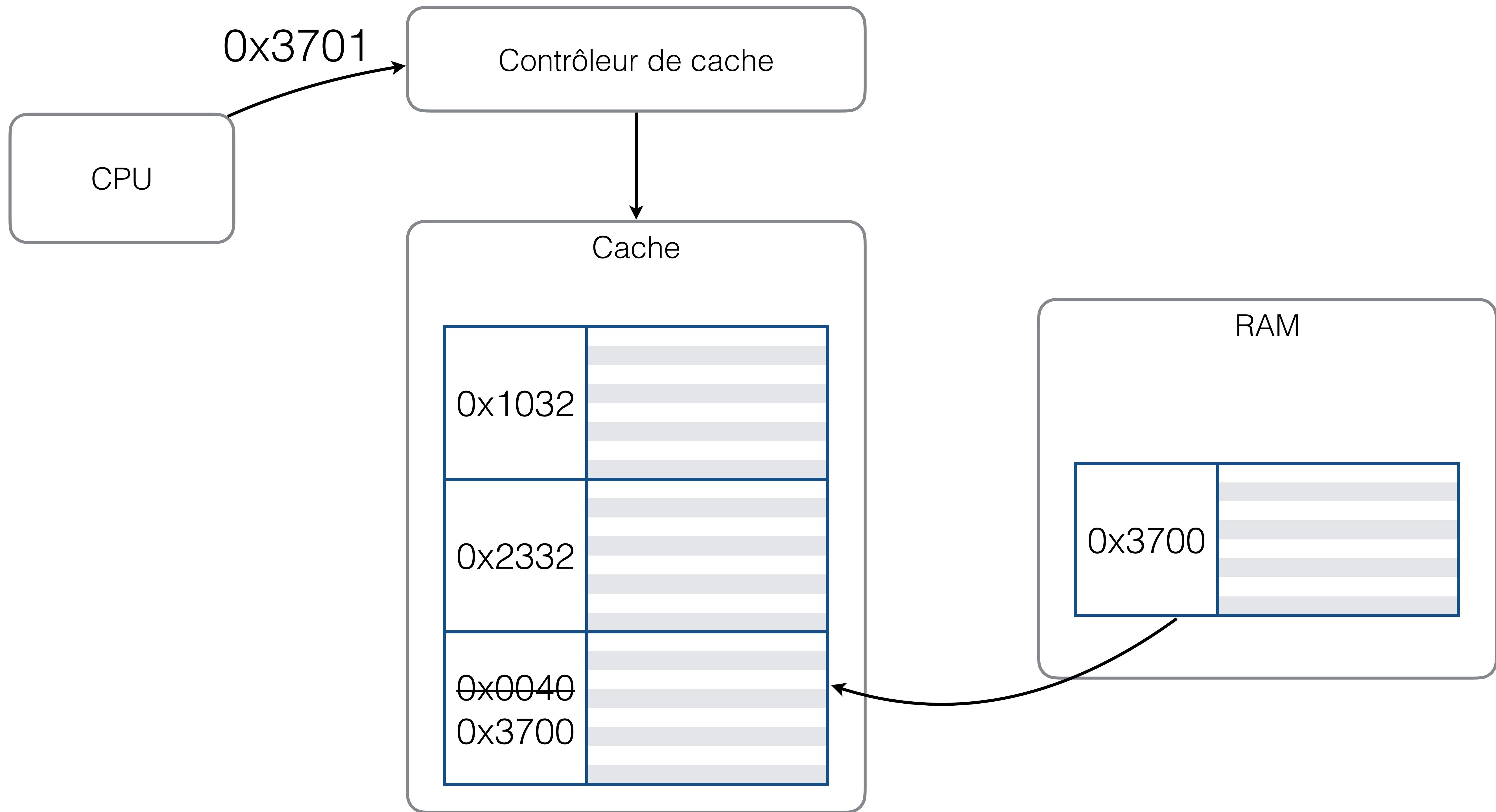
Si l'adresse est **absente** en cache,
on dit qu'il y a un *miss*.



Cache miss

- Lorsqu’il y a un *cache miss*, il faut effectuer les opérations suivantes:
  - trouver un bloc à remplacer
  - copier les données de la RAM vers la cache
- Quel bloc remplacer?
  - Prendre le bloc qui a été utilisé le moins récemment, *Least Recently Used* (*LRU*)

Lecture en cache



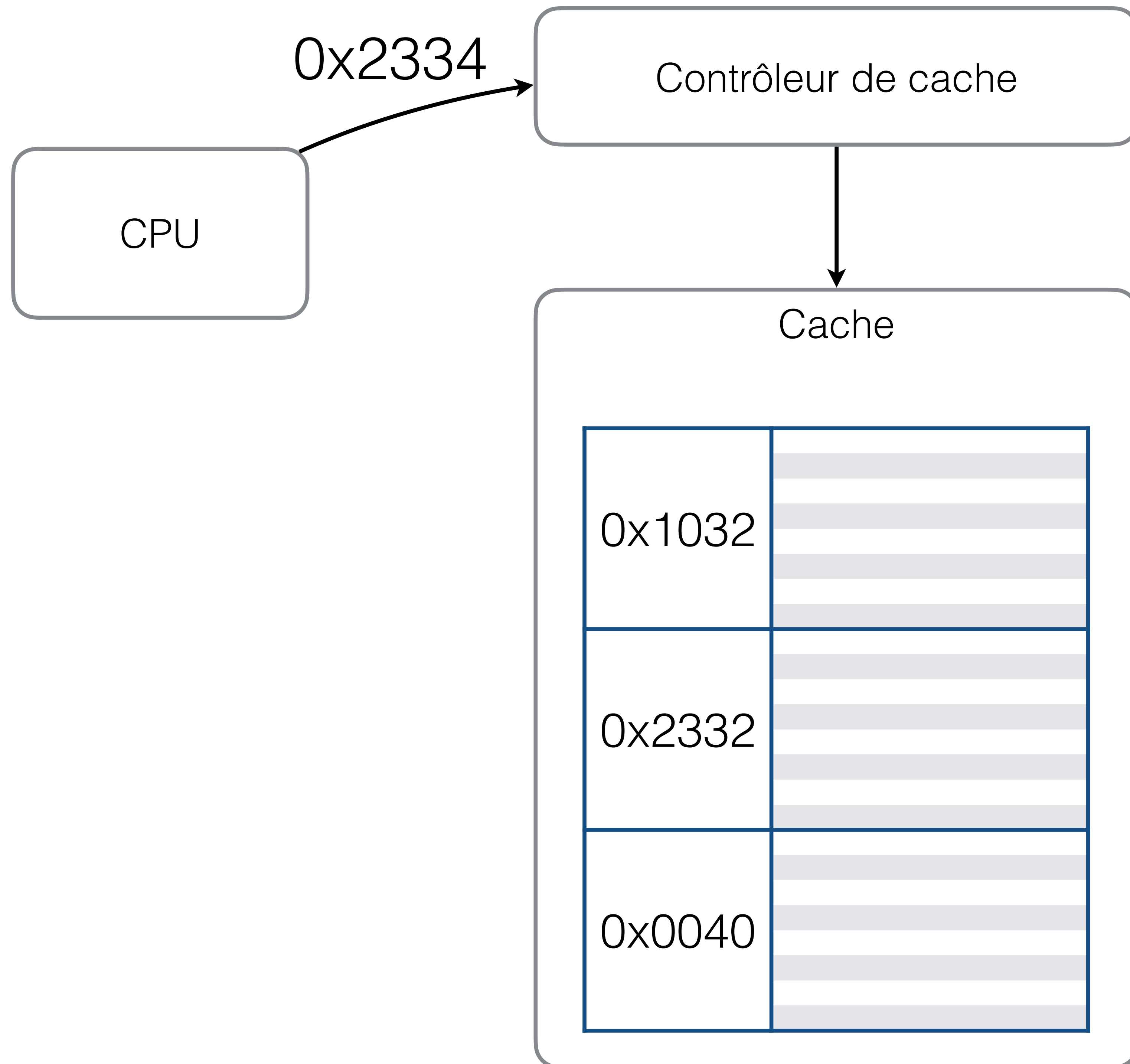
Cache *hit* vs *miss*

- Lorsqu'il veut une donnée, le microprocesseur cherche d'abord dans la cache.
  - Si la donnée s'y trouve, nous avons un *hit*.
  - Si la donnée ne s'y trouve pas, nous avons un *miss*
    - La donnée est transférée de la mémoire vers la cache. Les données adjacentes sont également transférées.
- Le *hit ratio* est la probabilité de retrouver une donnée dans la mémoire cache.

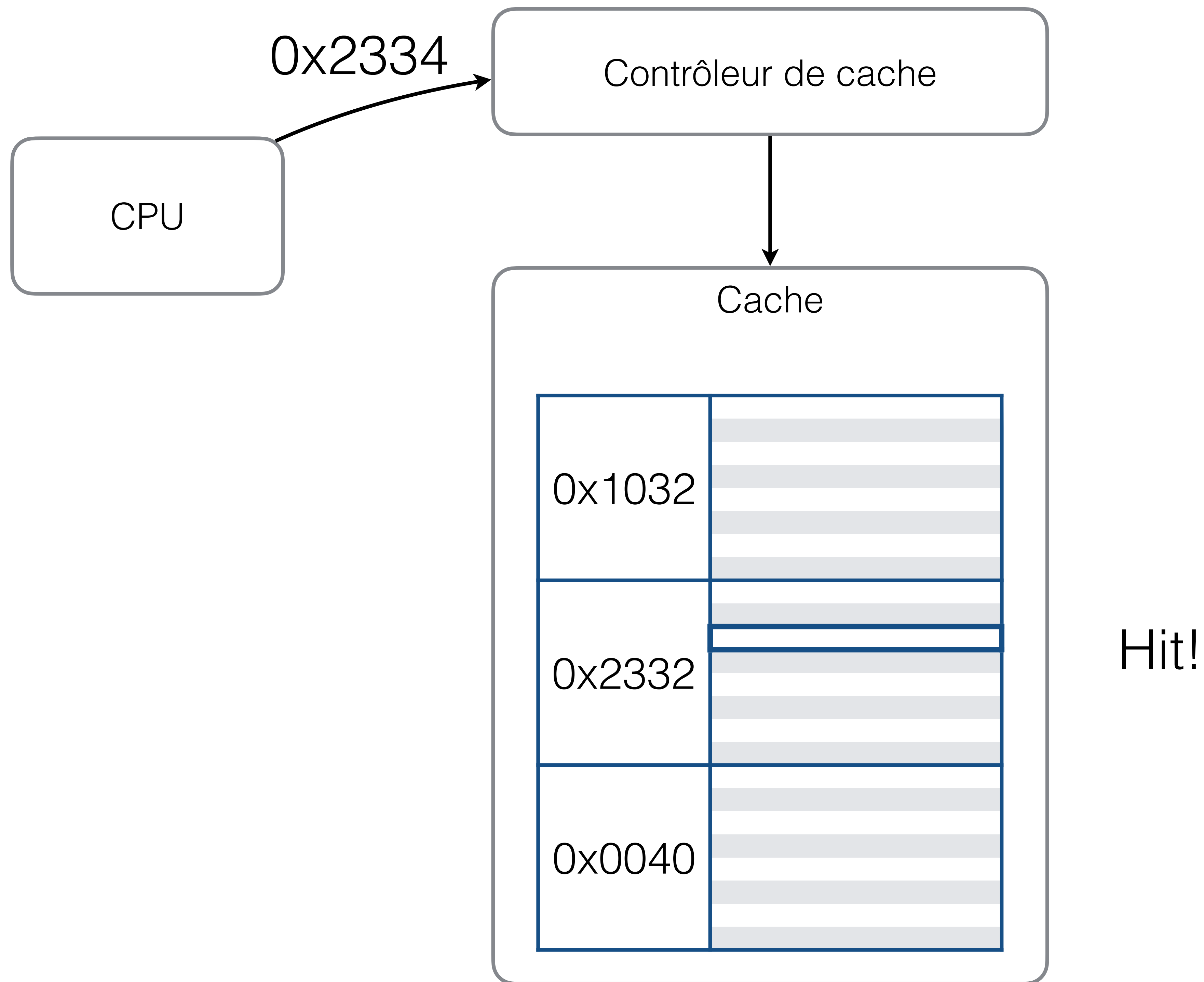
Écriture en cache

- Deux options lorsque l'on veut écrire en cache:
  1. ***Write-through*** : écrire les changements dans la RAM au fur et à mesure
     - On s'assure que la cache contienne **toujours** le même contenu que la RAM

Écriture en cache *write-through*

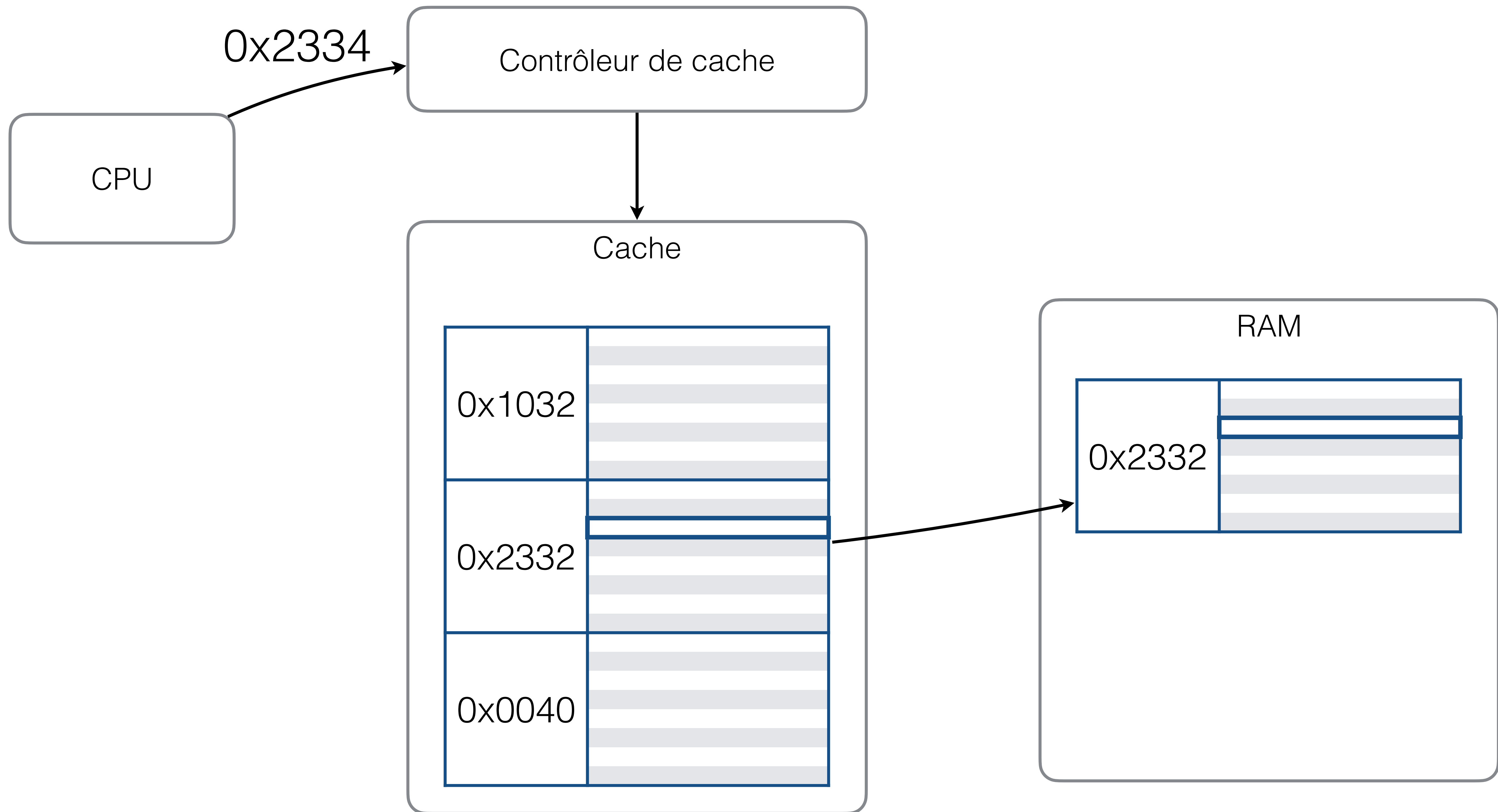


Écriture en cache *write-through*

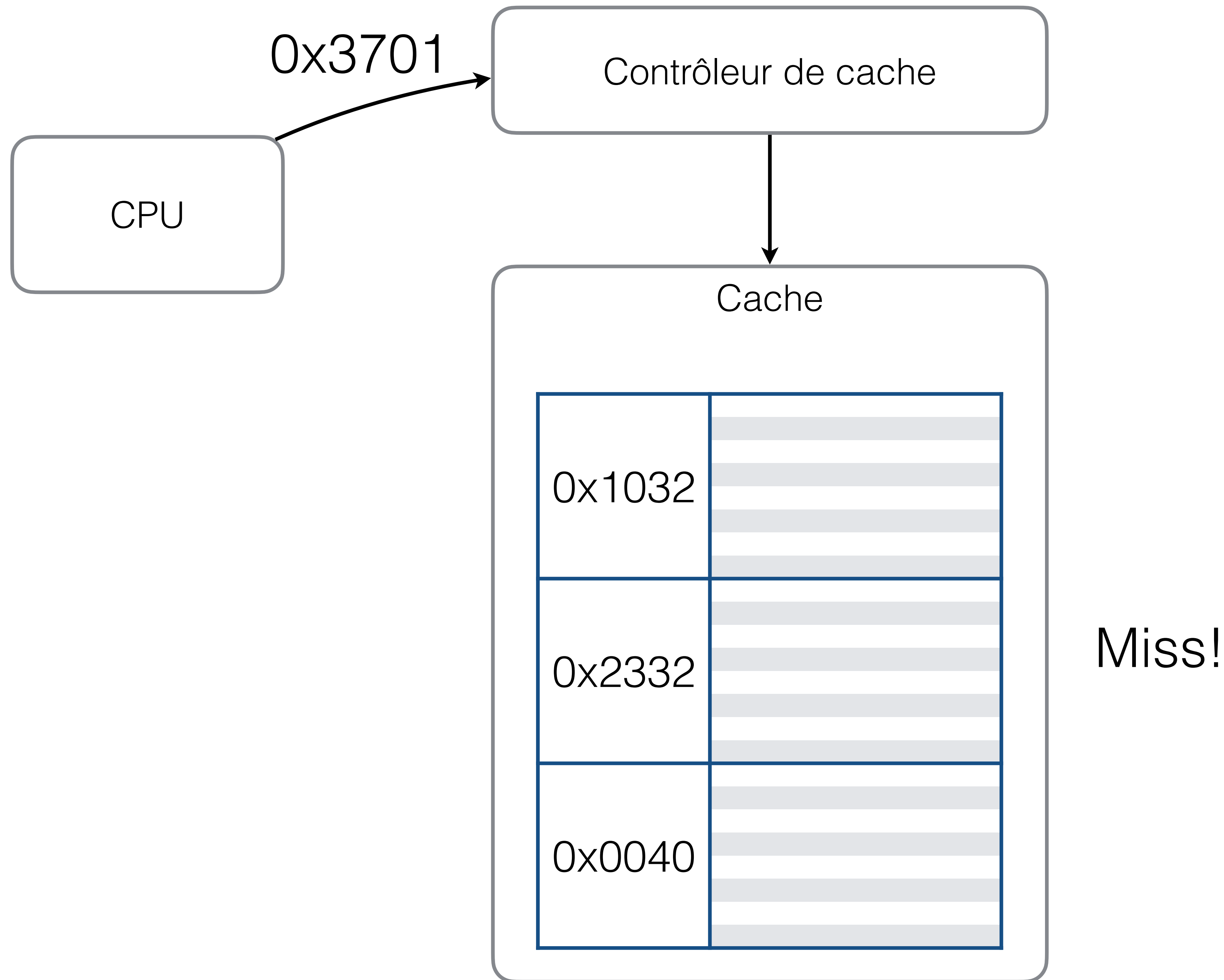


Hit!

Écriture en cache *write-through*

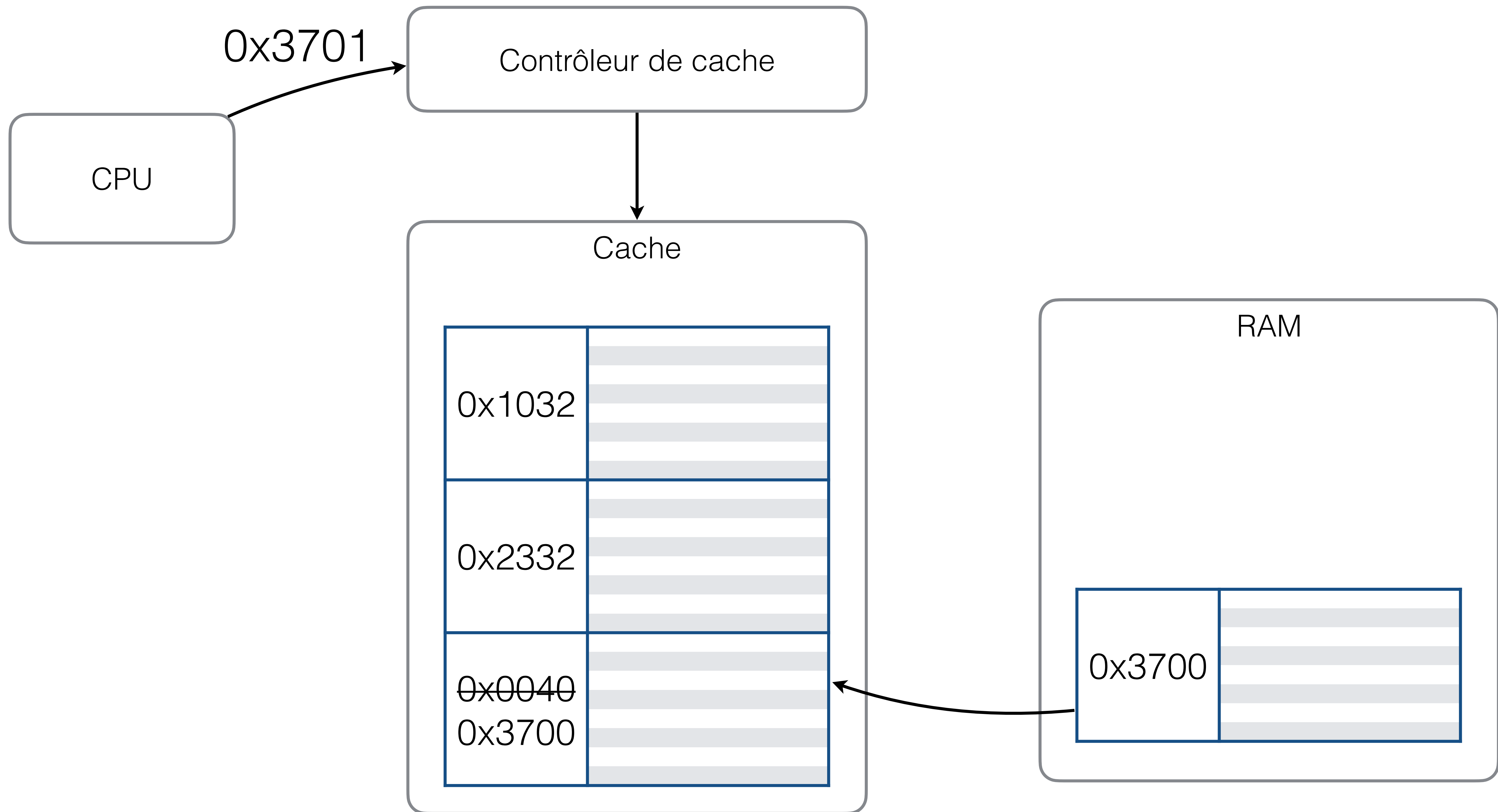


Écriture en cache *write-through*

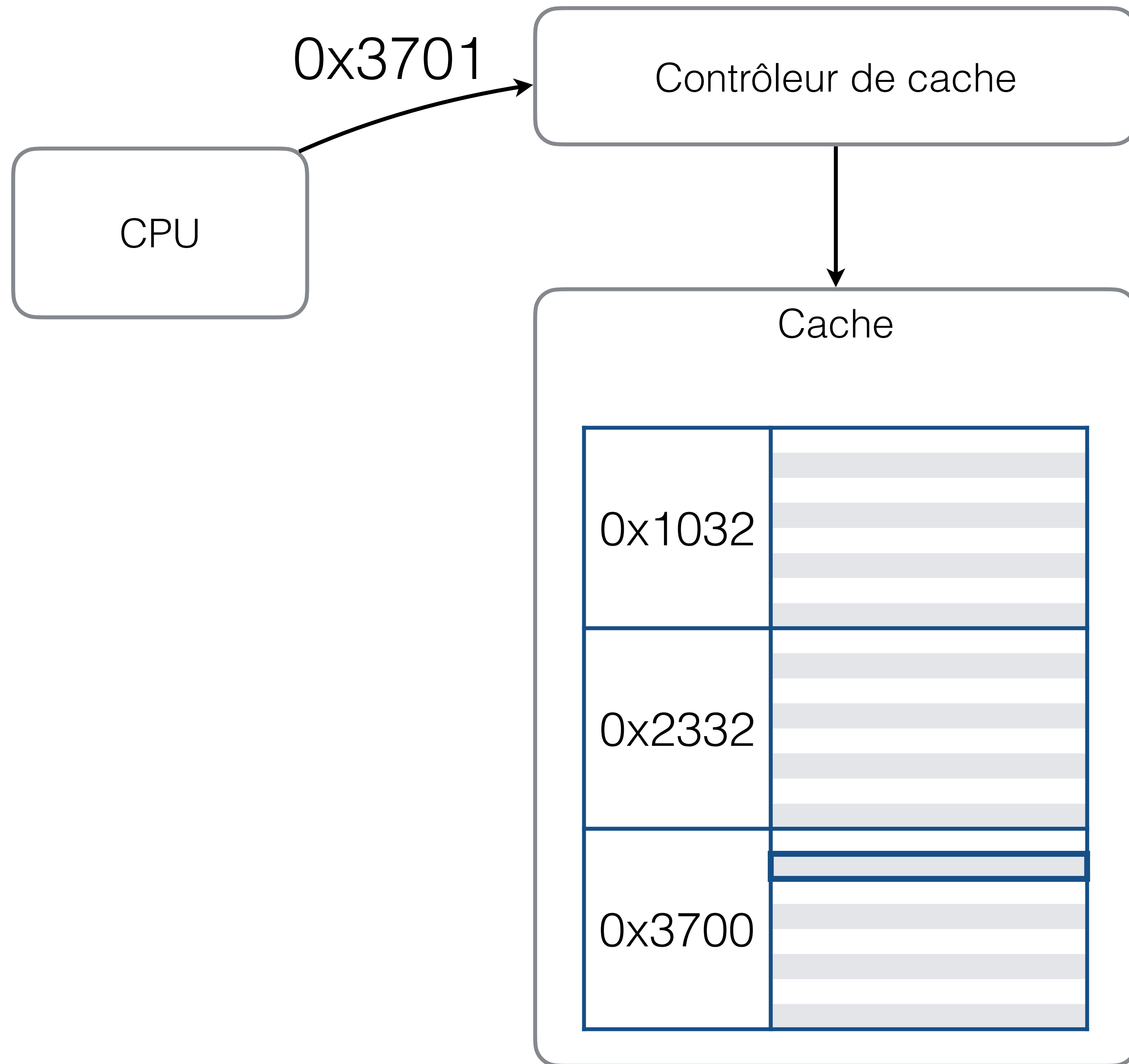


Miss!

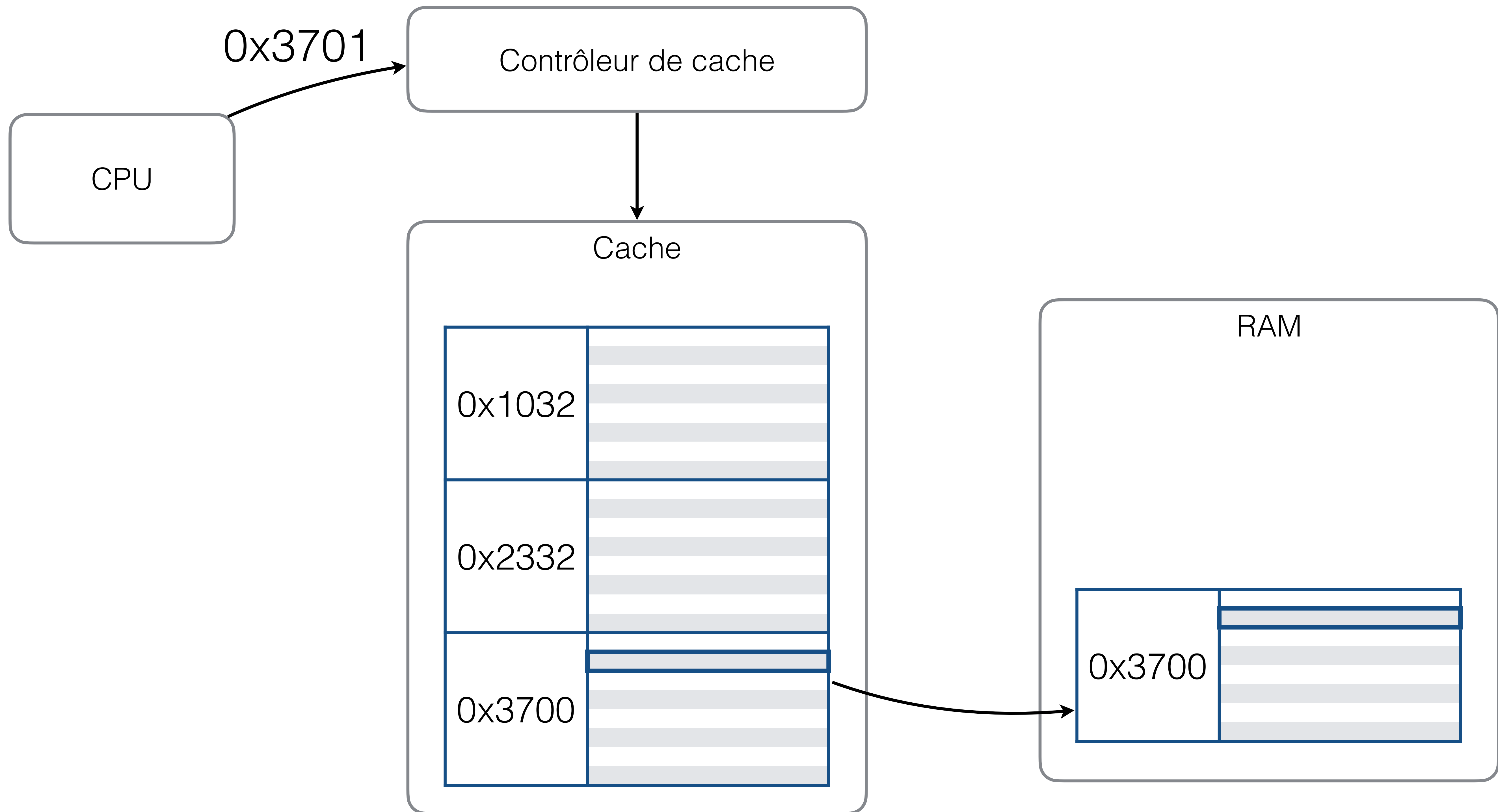
Écriture en cache *write-through*



Écriture en cache *write-through*

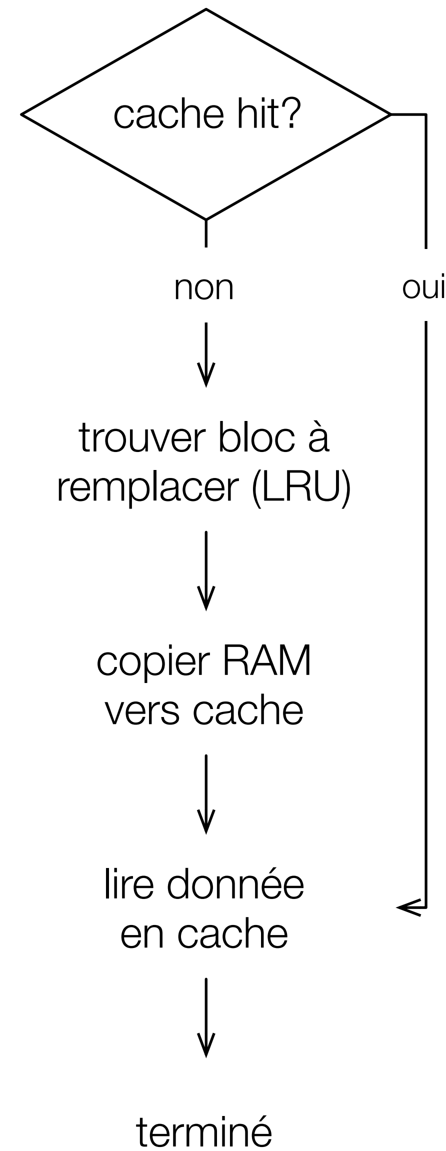


Écriture en cache *write-through*

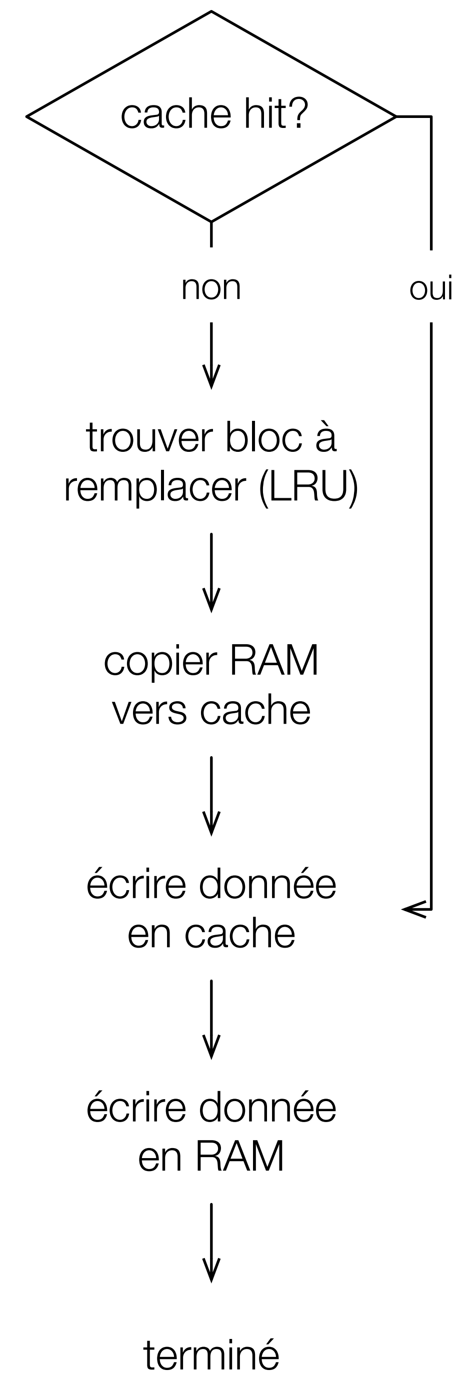


Cache *write-through*

Lecture



Écriture

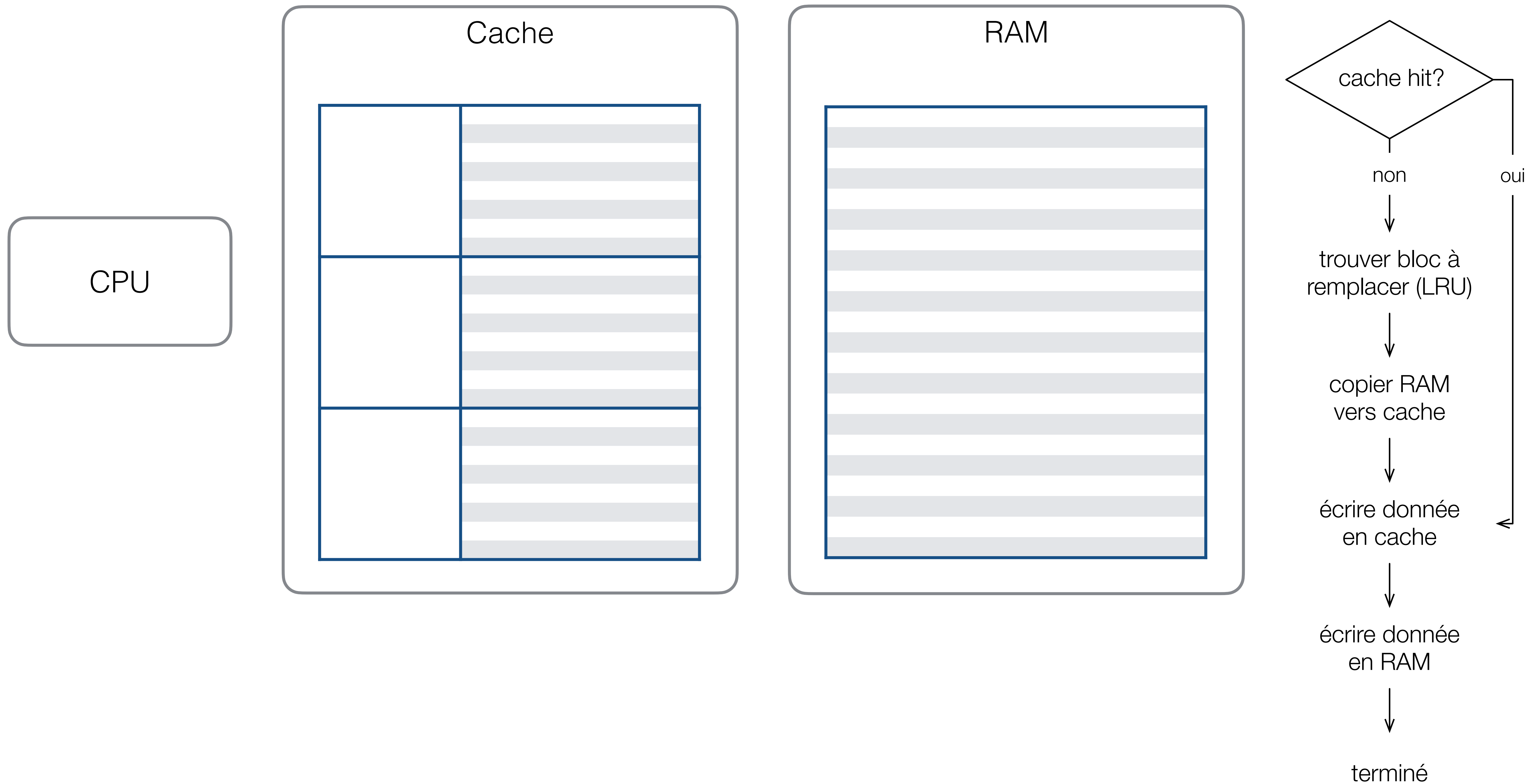


Exercice

- Décrivez les étapes nécessaires pour que le micro-processeur **écrive** une donnée en mémoire si:
  - l’adresse mémoire **n’est pas** dans la cache;
  - le système utilise la stratégie ***write-through***.

Exercice

- Décrivez les étapes nécessaires pour que le micro-processeur **écrive** une donnée en mémoire si:
  - l'adresse mémoire **n'est pas** dans la cache;
  - le système utilise la stratégie ***write-through***.



Exercice—solution

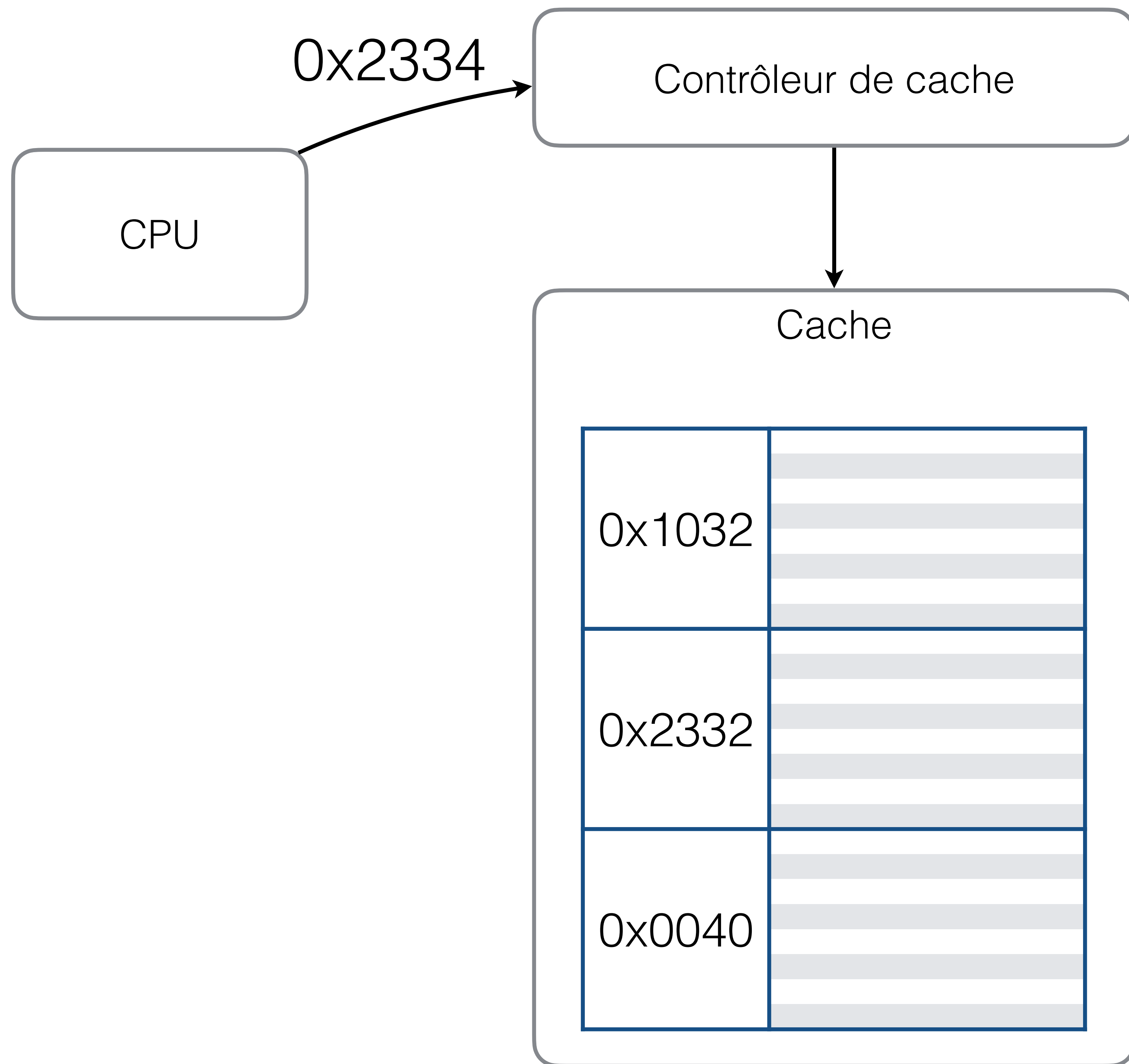
- Étapes:
  - « miss » en cache
  - trouver bloc à remplacer
  - copier bloc RAM vers cache
  - écriture en cache
  - écriture en RAM

Écriture en cache

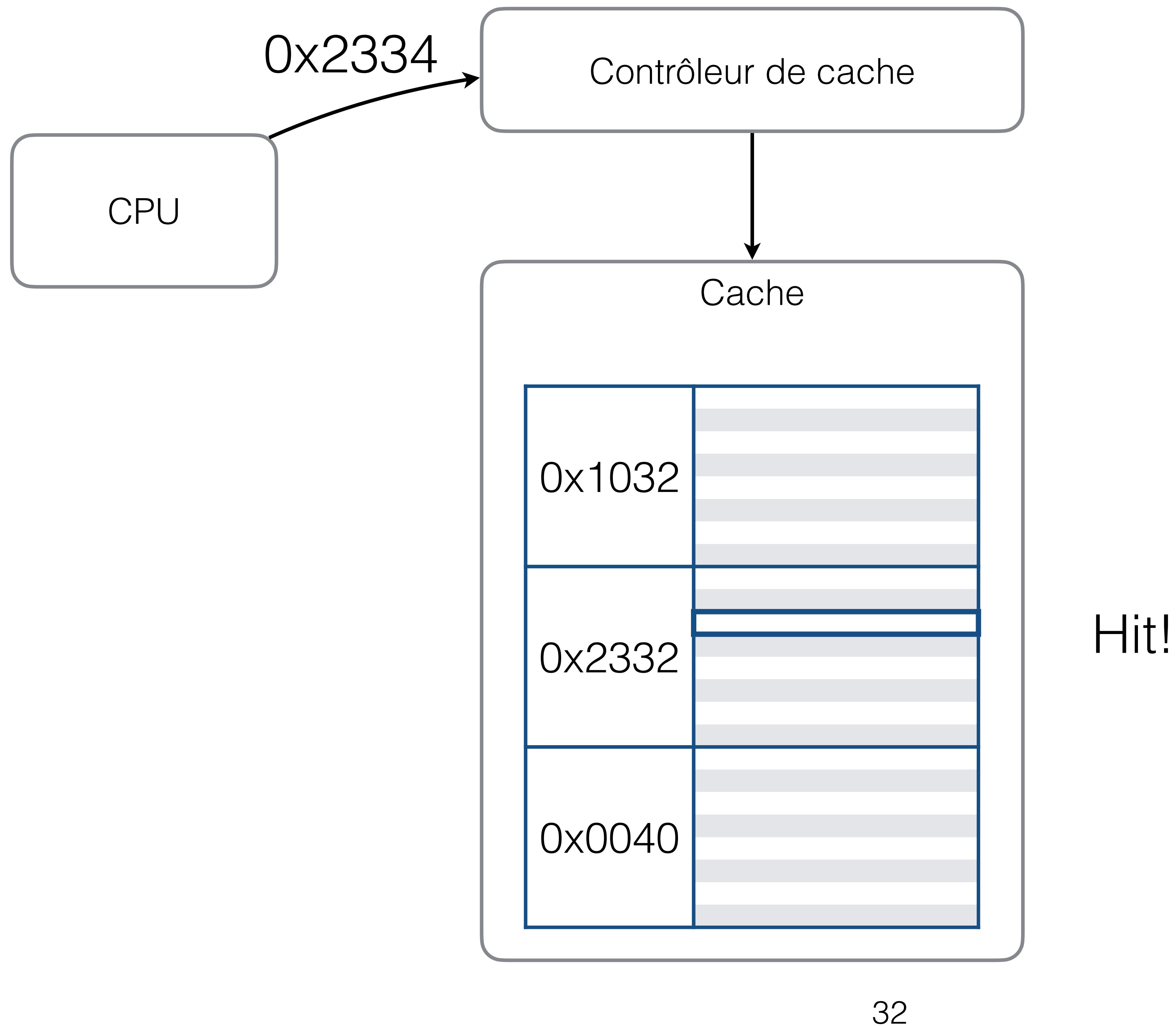
- Deux options lorsque l'on veut écrire en cache:

1. *Write-through* : écrire les changements dans la RAM au fur et à mesure
    - On s'assure que la cache contienne **toujours** le même contenu que la RAM
2. ***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
    - La RAM est mise à jour seulement quand c'est nécessaire

Écriture en cache *write-back*



Écriture en cache *write-back*



Hit!

Écriture en cache *write-back*

0x2334

CPU

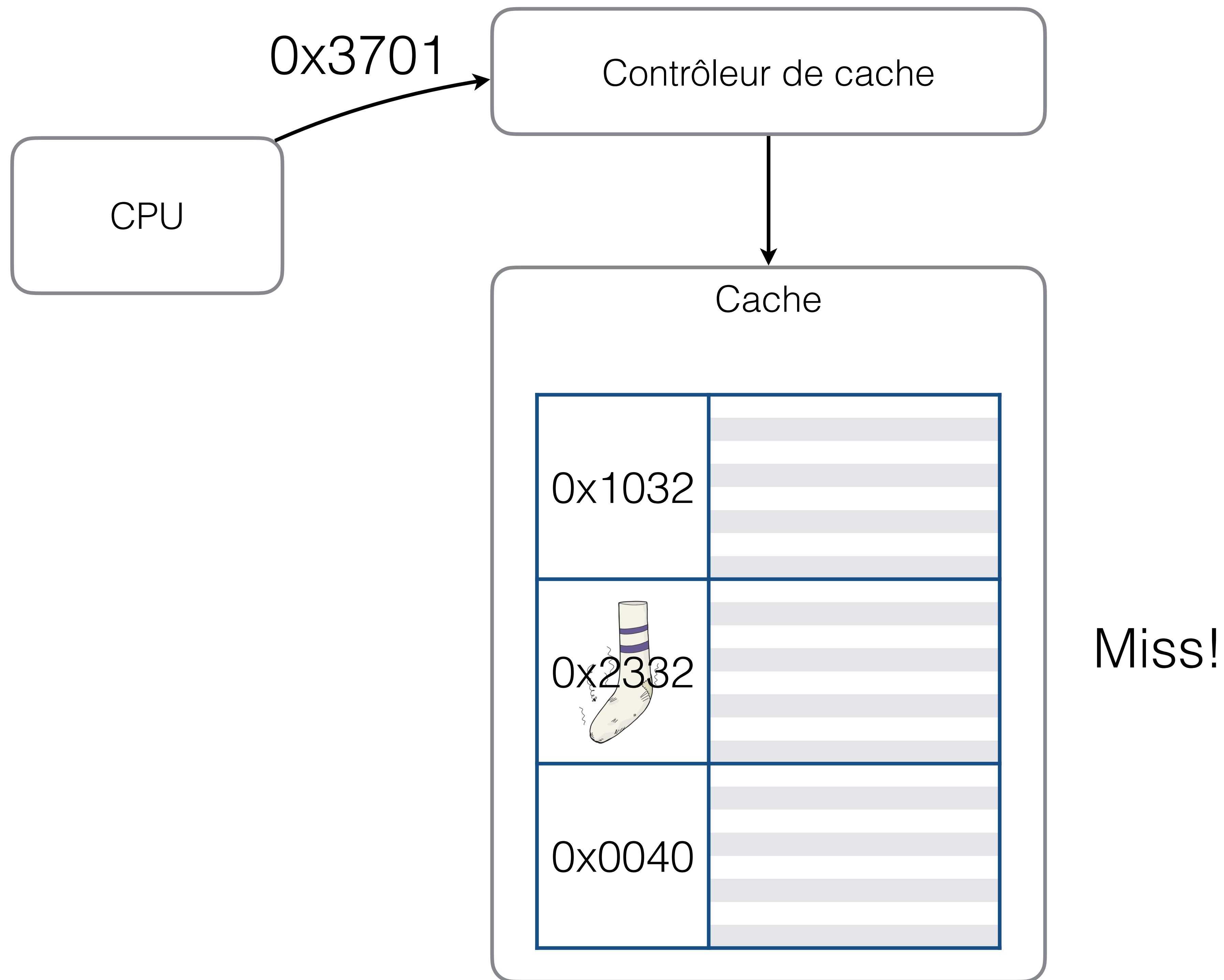
Contrôleur de cache

Cache

0x1032	
0x2332	
0x0040	

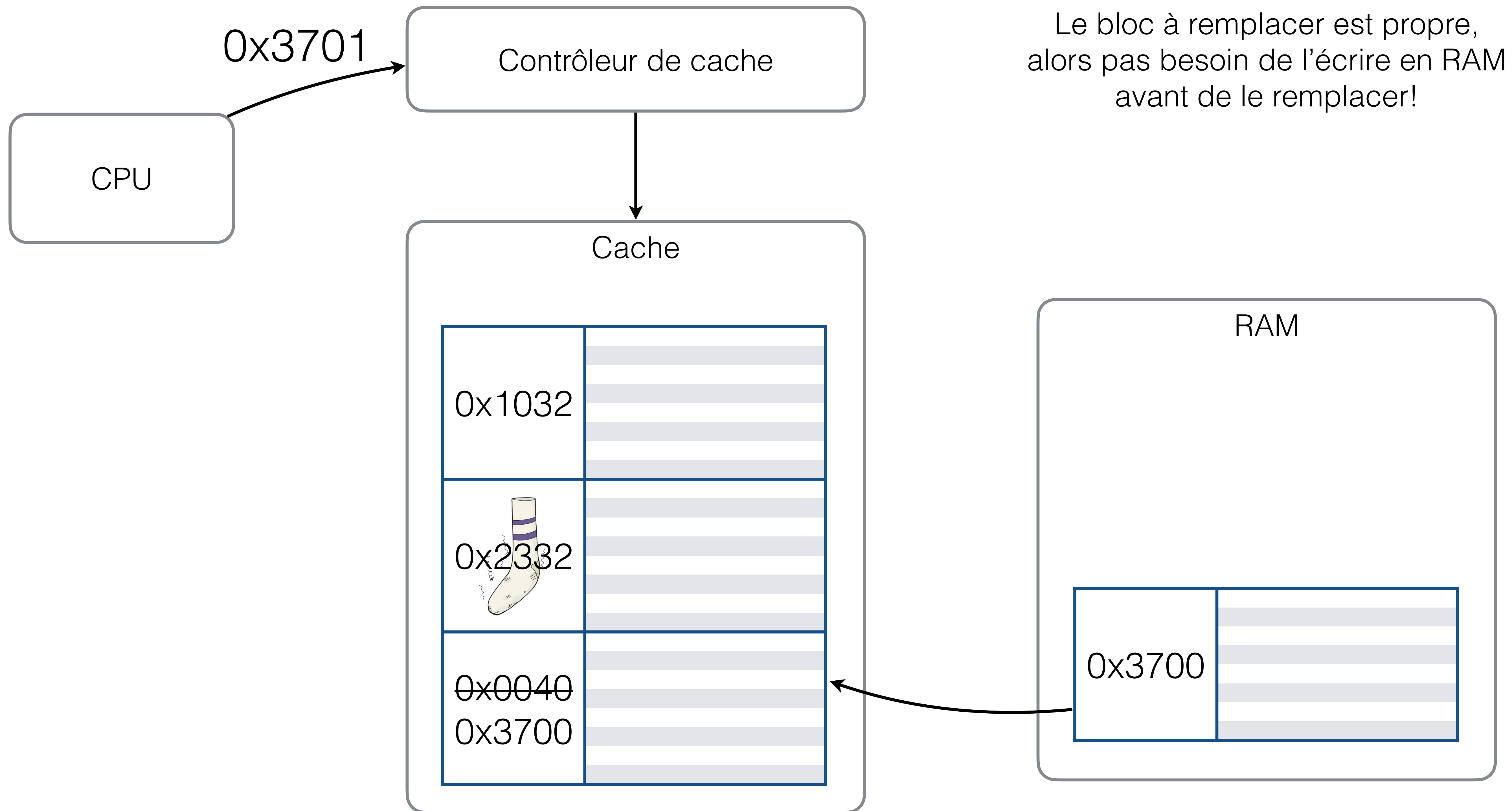
Le bloc est sale (« dirty »)!

Écriture en cache *write-back*

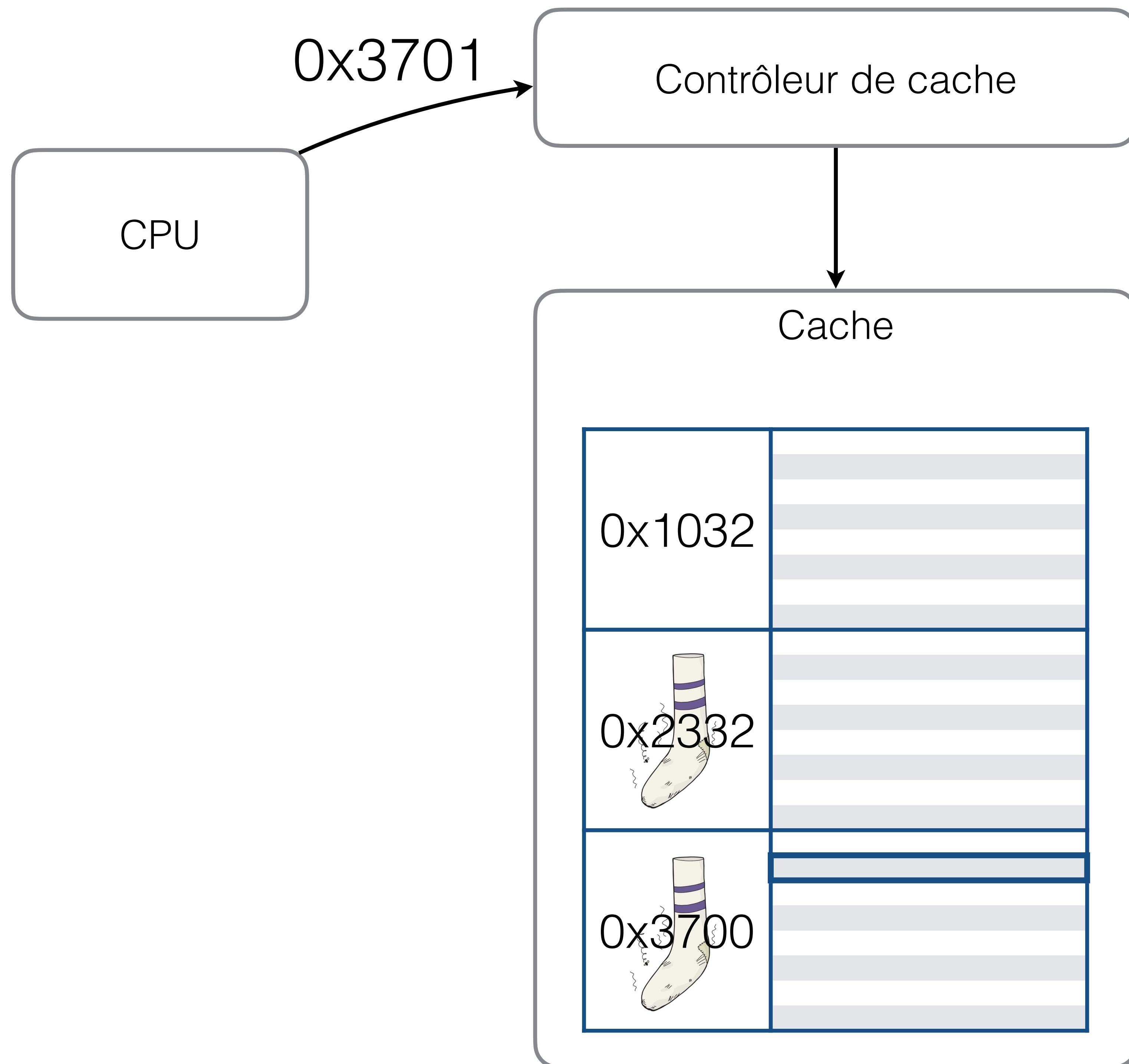


Miss!

Écriture en cache *write-back*

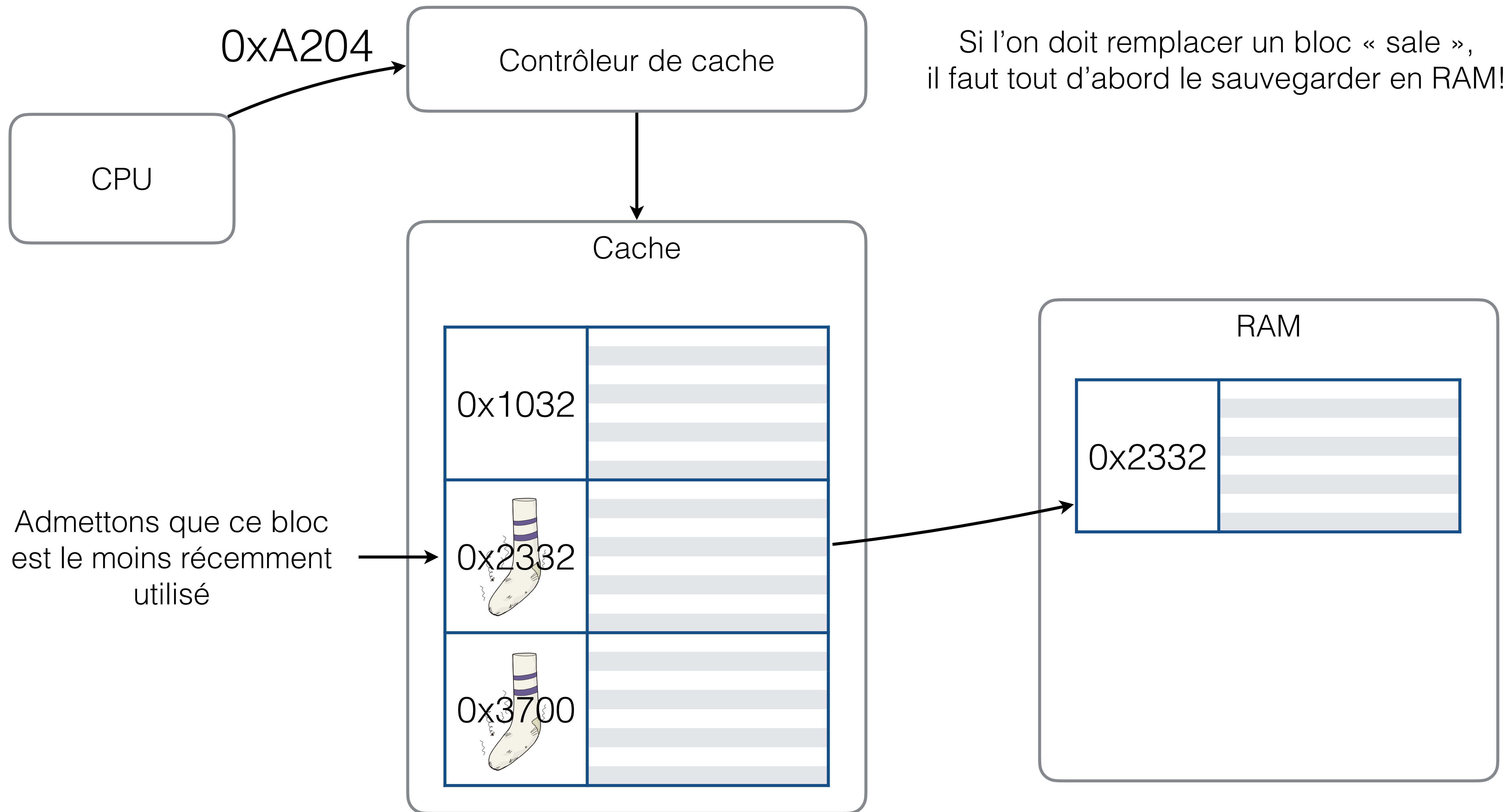


Écriture en cache *write-back*

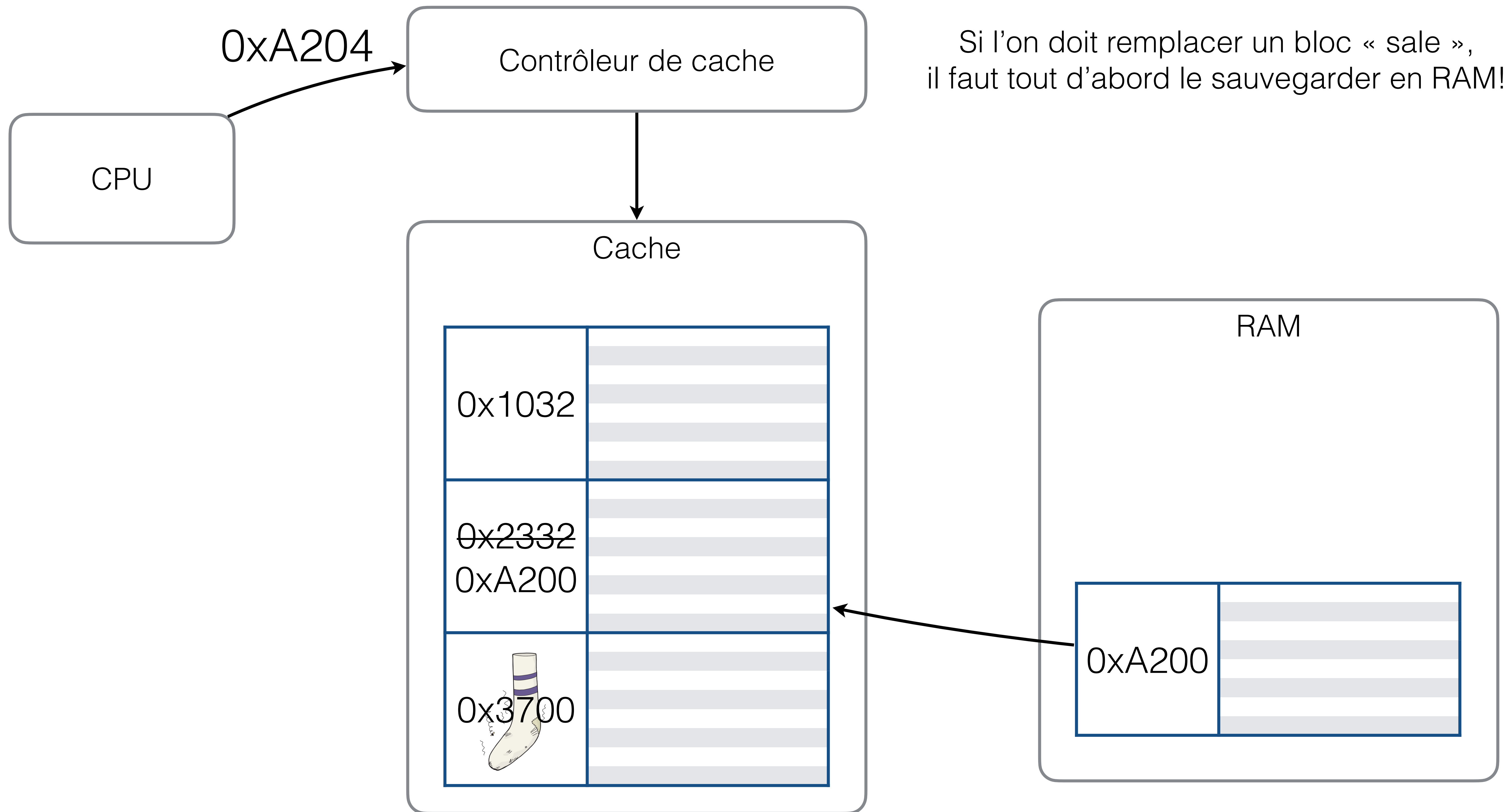


On peut ensuite écrire dans le bloc en cache, et l'identifier comme « sale »

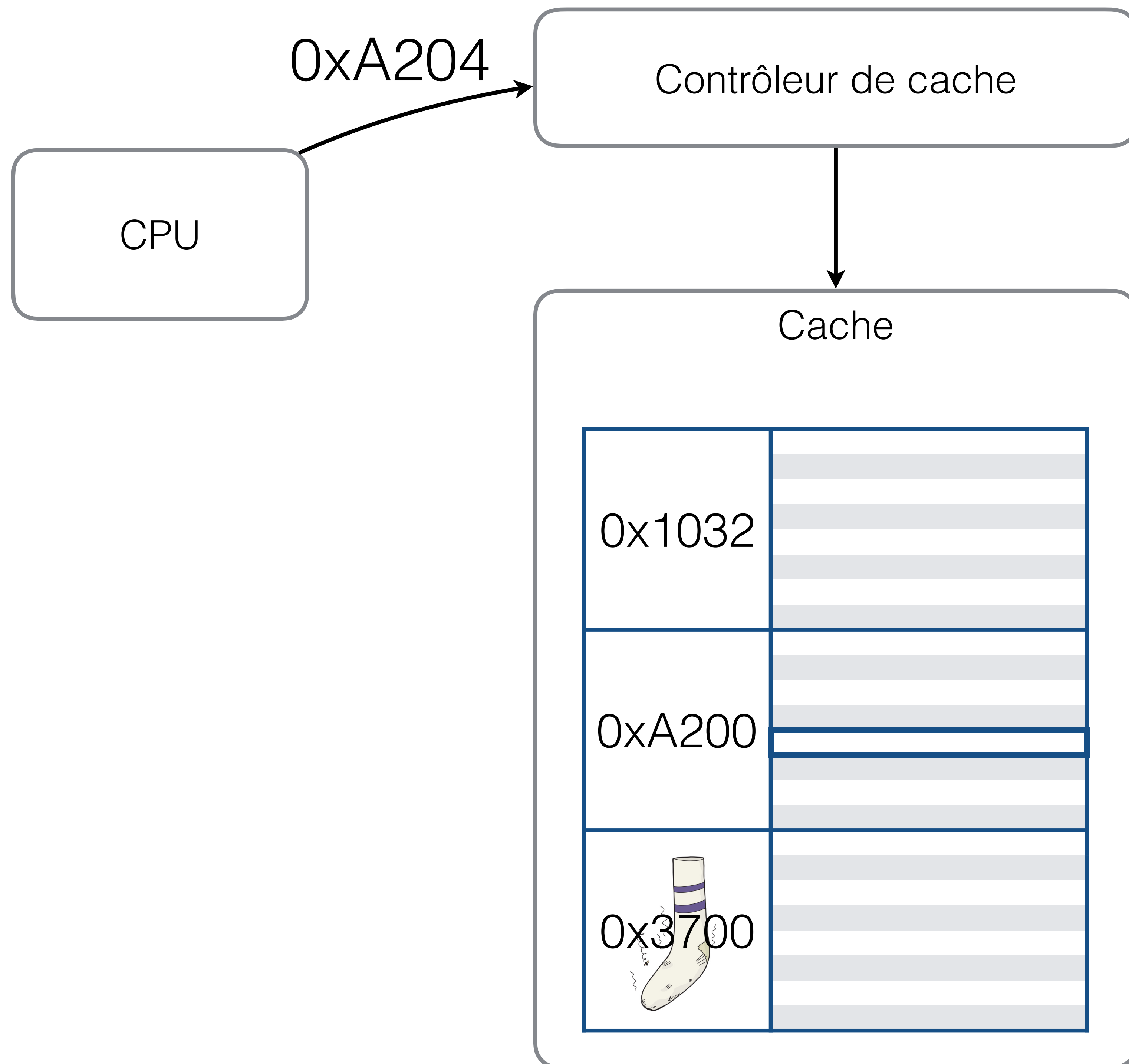
Écriture en cache *write-back*



Écriture en cache *write-back*



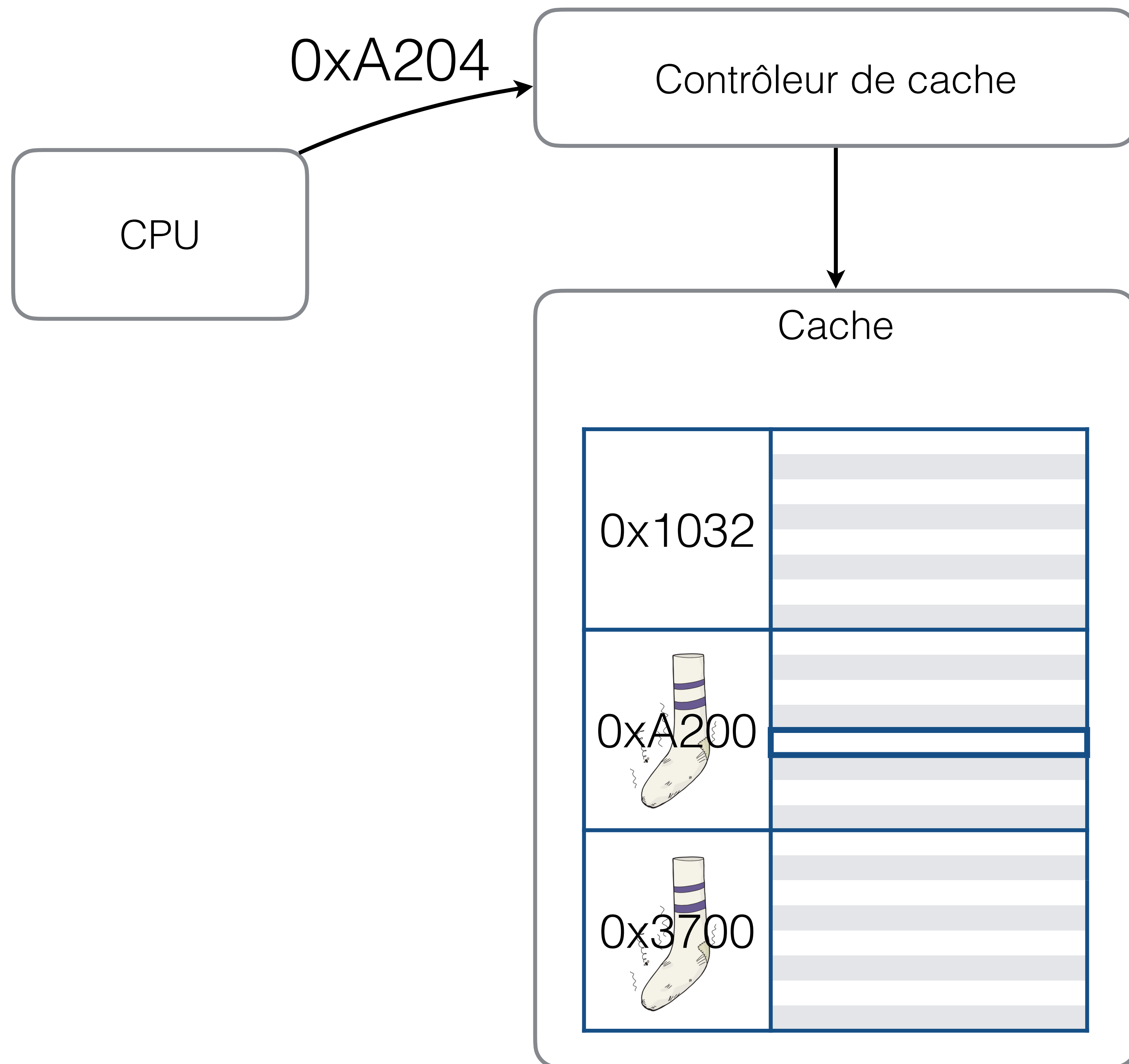
Écriture en cache *write-back*



Si l'on doit remplacer un bloc « sale »,
il faut tout d'abord le sauvegarder en RAM!

L'écriture peut être effectuée ensuite.

Écriture en cache *write-back*



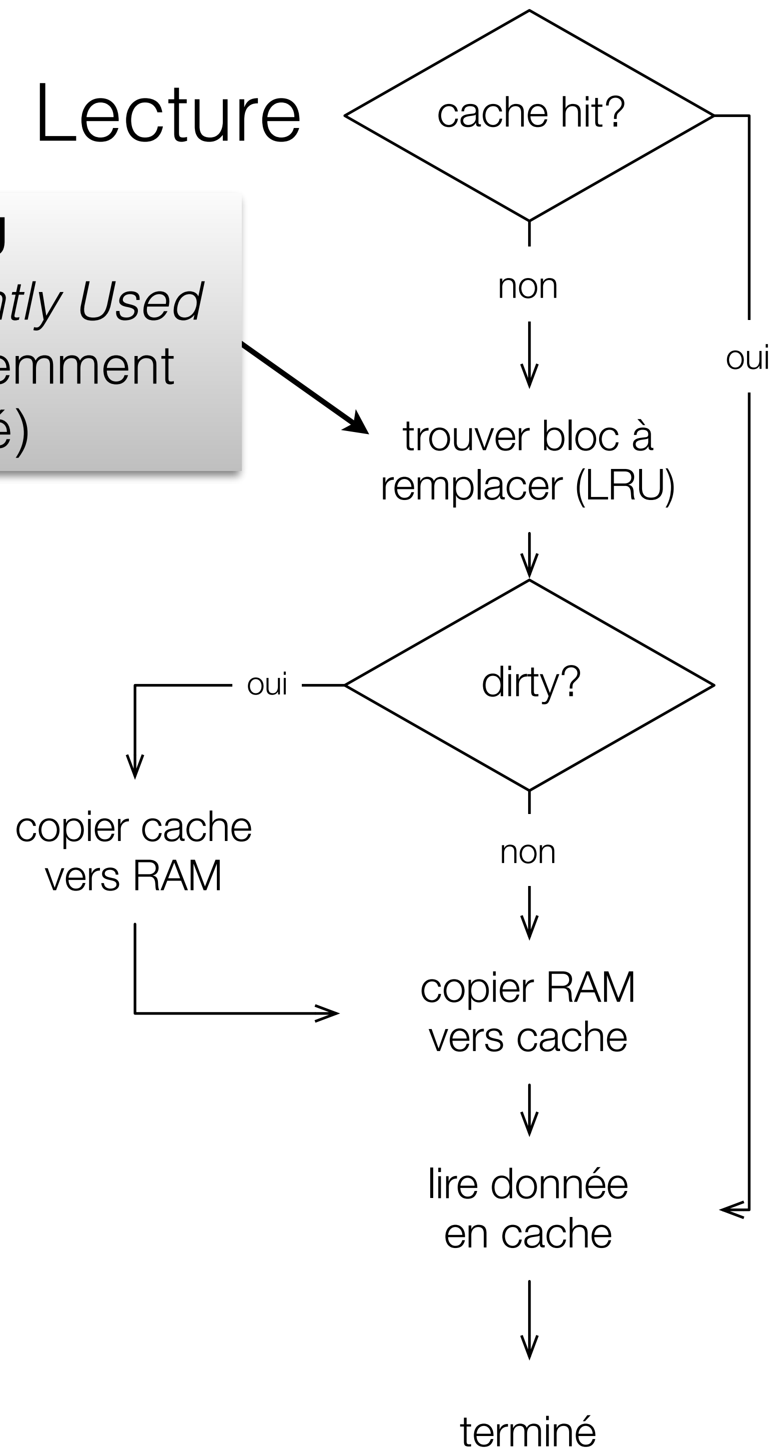
Si l’on doit remplacer un bloc « sale »,
il faut tout d’abord le sauvegarder en RAM!

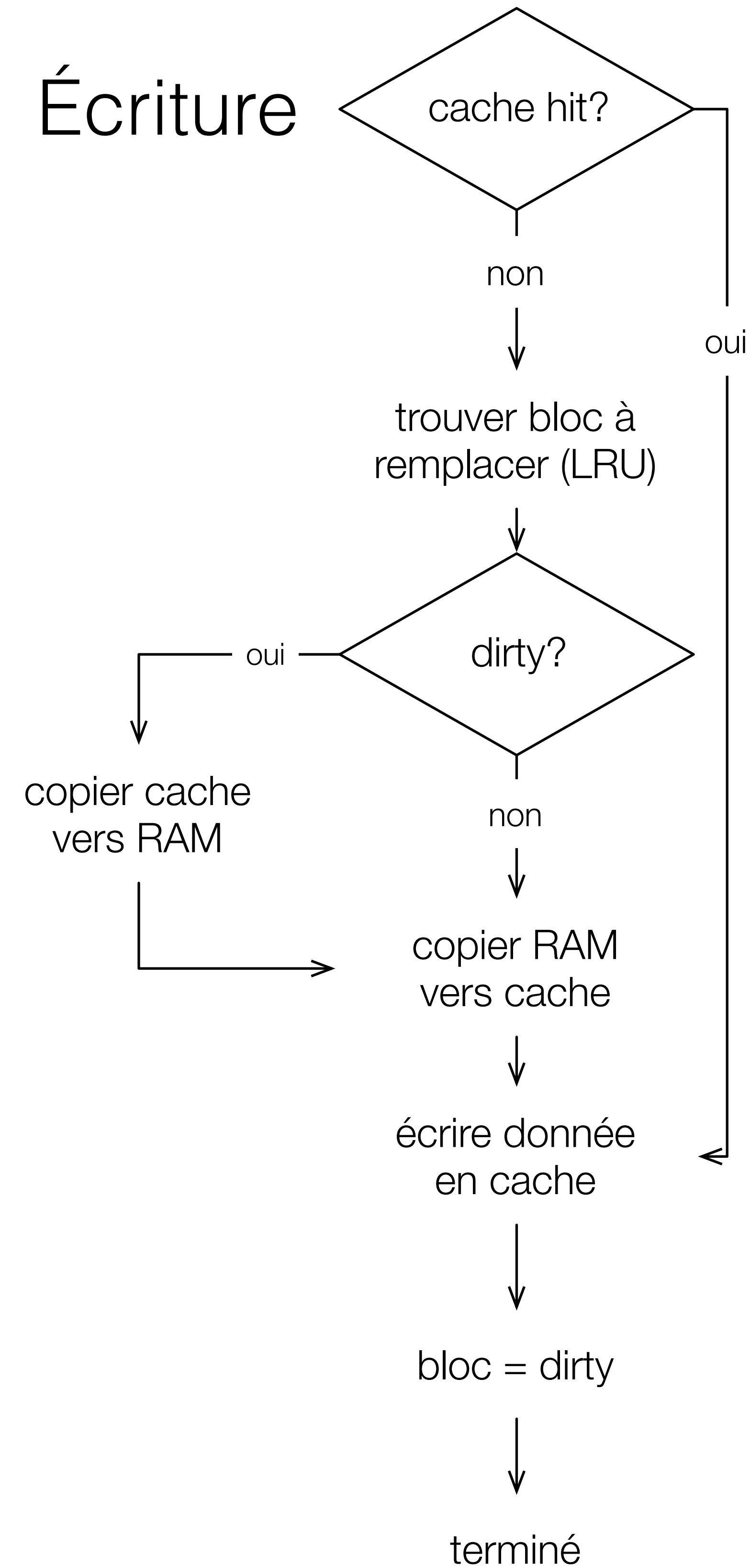
L’écriture peut être effectuée ensuite.

Et le bloc est identifié comme « sale »

Cache *write-back*

LRU
Least Recently Used
(moins récemment utilisé)





Récapitulation

Write-through

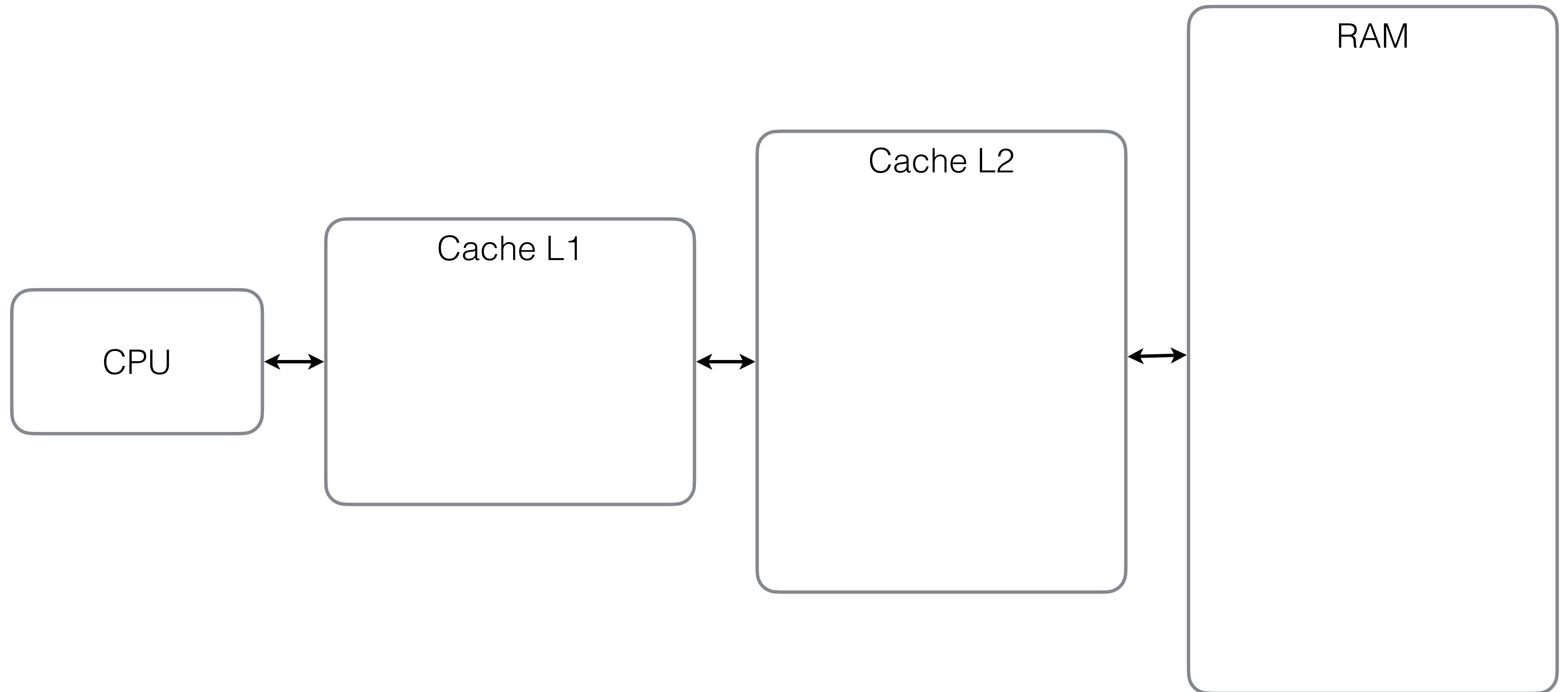
Write-back

Mets la RAM à jour
à chaque écriture

Hiérarchie de caches

- En pratique, les microprocesseurs modernes utilisent **plusieurs** mémoires cache.
- On progresse de plus petit (rapide) vers plus grande (lente)

Hiérarchie de caches



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 (ex: 64Ko par coeur pour les Intel i7)
- 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.

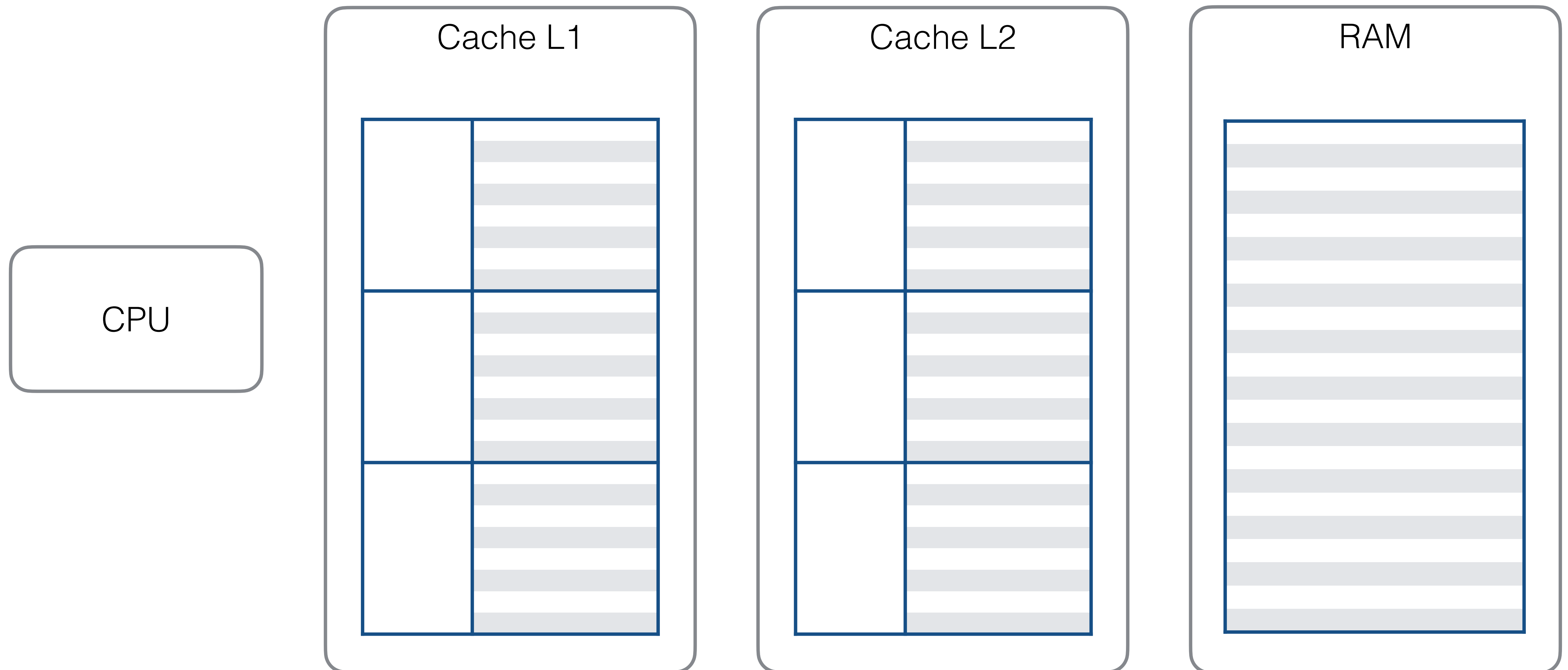
- L2 et L3 sont moins rapide que L1, mais environ 10 fois plus rapide que la mémoire.

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***.

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***.



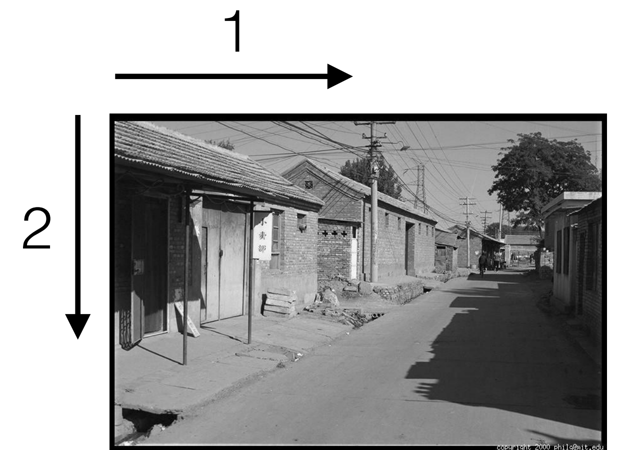
Exercice — solution

- Étapes:
    - « miss » en cache L1
    - trouver bloc à remplacer en cache L1
    - copier bloc de cache L2 à L1
        - « miss » en cache L2
        - trouver bloc à remplacer en cache L2
        - copier bloc de RAM à L2
    - écriture en cache L1
    - écriture en cache L2
        - écriture en RAM

Retour

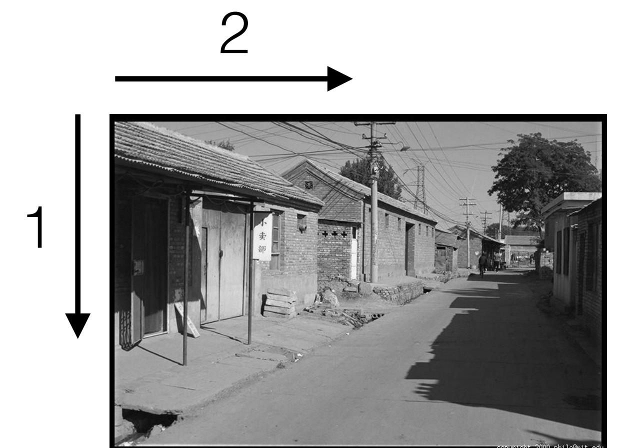
- Laquelle de ces deux versions est la plus rapide?

```
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;
   }
 }
```



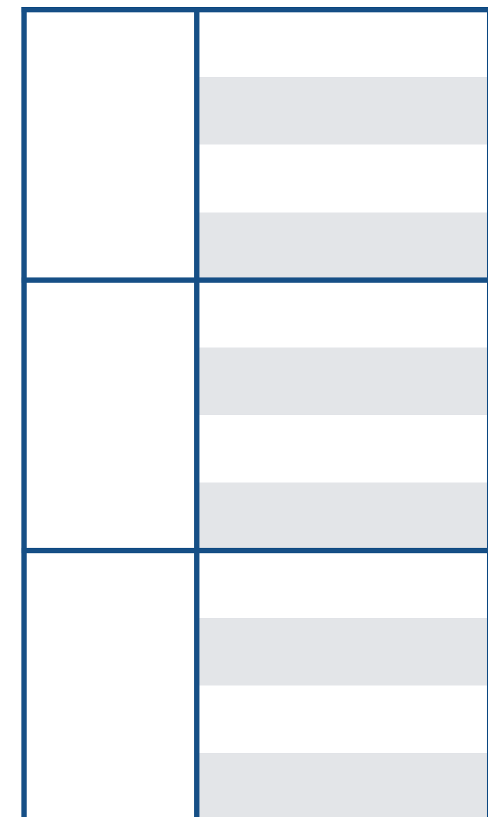
i = lignes (“y”)
j = colonnes (“x”)

Ordre d'accès mémoire

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Exemple
Une cache de 3 blocs possédant chacun 4 mots



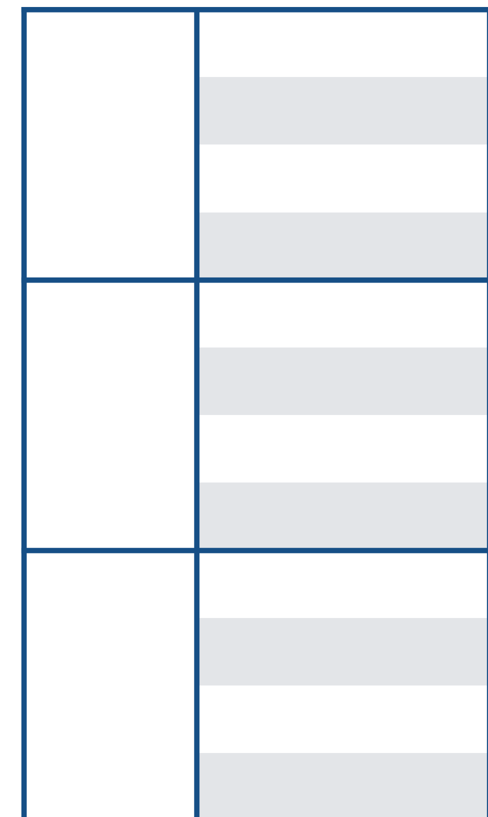
```
for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 4; j++) {
    img[i*4 + j] = 0;
  }
}
```

Ordre d’accès mémoire

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

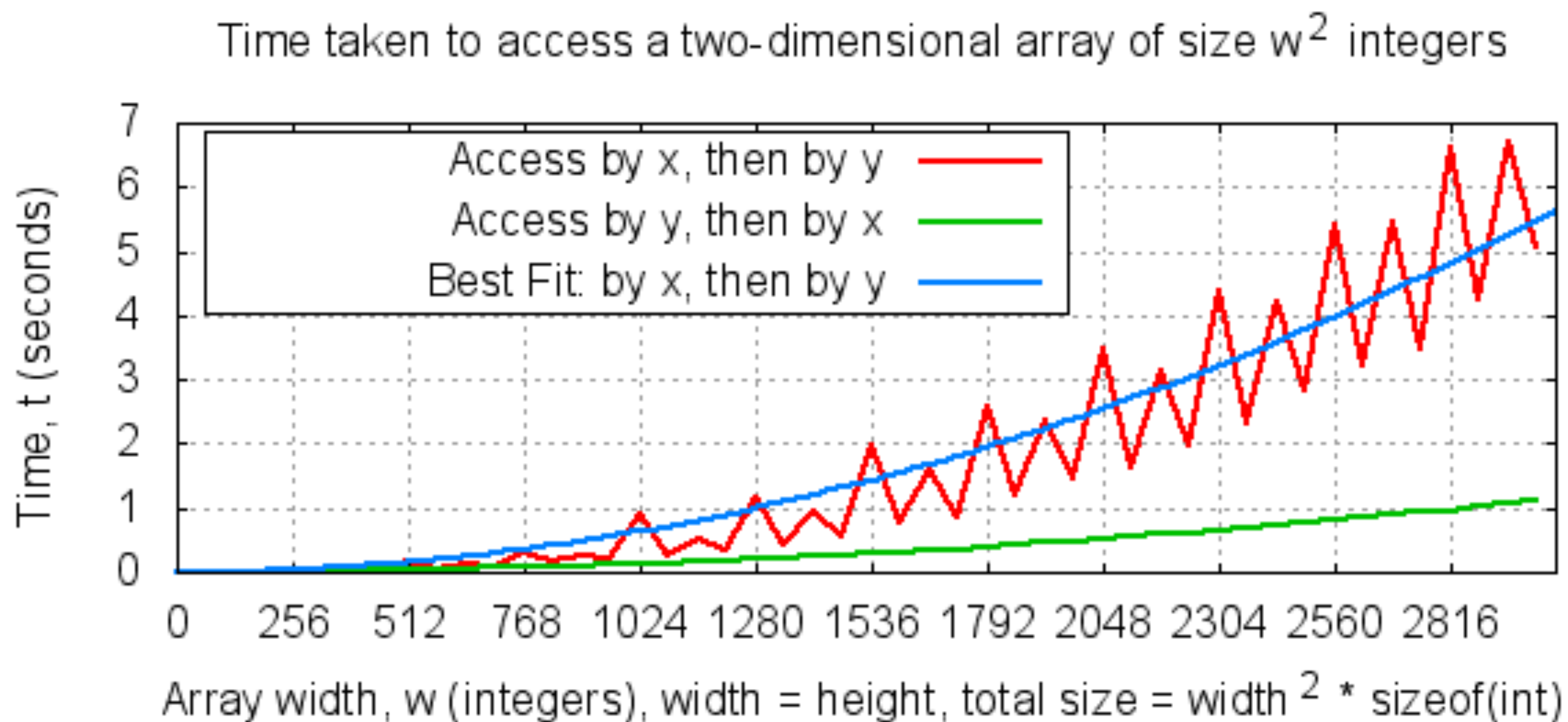
Exemple
Une cache de 3 blocs
possédant chacun 4 mots

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



```
for (int j = 0; j < 4; j++) {
  for (int i = 0; i < 4; i++) {
    img[i*4 + j] = 0;
  }
}
```

Ordre d'accès mémoire



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

Mémoire virtuelle vs cache

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

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 en mémoire paginée

- En mémoire paginée, il faut accéder à la table de pages afin de traduire l’adresse virtuelle en adresse physique.
- Comme la table de page est souvent volumineuse, elle est elle-même en mémoire
- Il faut donc faire **deux lectures** de la mémoire pour aller chercher une instruction, ce qui est très long!

TLB: *Translation Lookaside Buffer*

- Le TLB (*Translation Lookaside Buffer*), est un groupe de registres très rapides à l'intérieur du CPU.
- Le MMU s'en sert comme une **cache** de la table des pages.
- Pour traduire une adresse virtuelle, le MMU regarde d'abord si la page à traduire est dans le TLB.
  - Puisque le TLB contient les dernières entrées de la table des pages utilisées, on 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%!