Architecture des ordinateurs II

Les mémoires caches

- 1 Objectif et principe d'une mémoire cache
- 2 Où placer un bloc?
- 3 Comment un bloc est-il trouvé?
- 4 Quel bloc remplacé lors d'un défaut?
- 5 Comment sont traitées les écritures?

Objectifs et principes du cache

- Principe de base du cache :
  - Les mots mémoires les plus fréquemment utilisés sont conservés dans une mémoire rapide (cache) plutôt que dans une mémoire lente (mémoire centrale).

Les mémoires doivent répondre à deux contraintes contradictoires :

- Taille importante
- Temps d'accès court

Objectifs et principes du cache

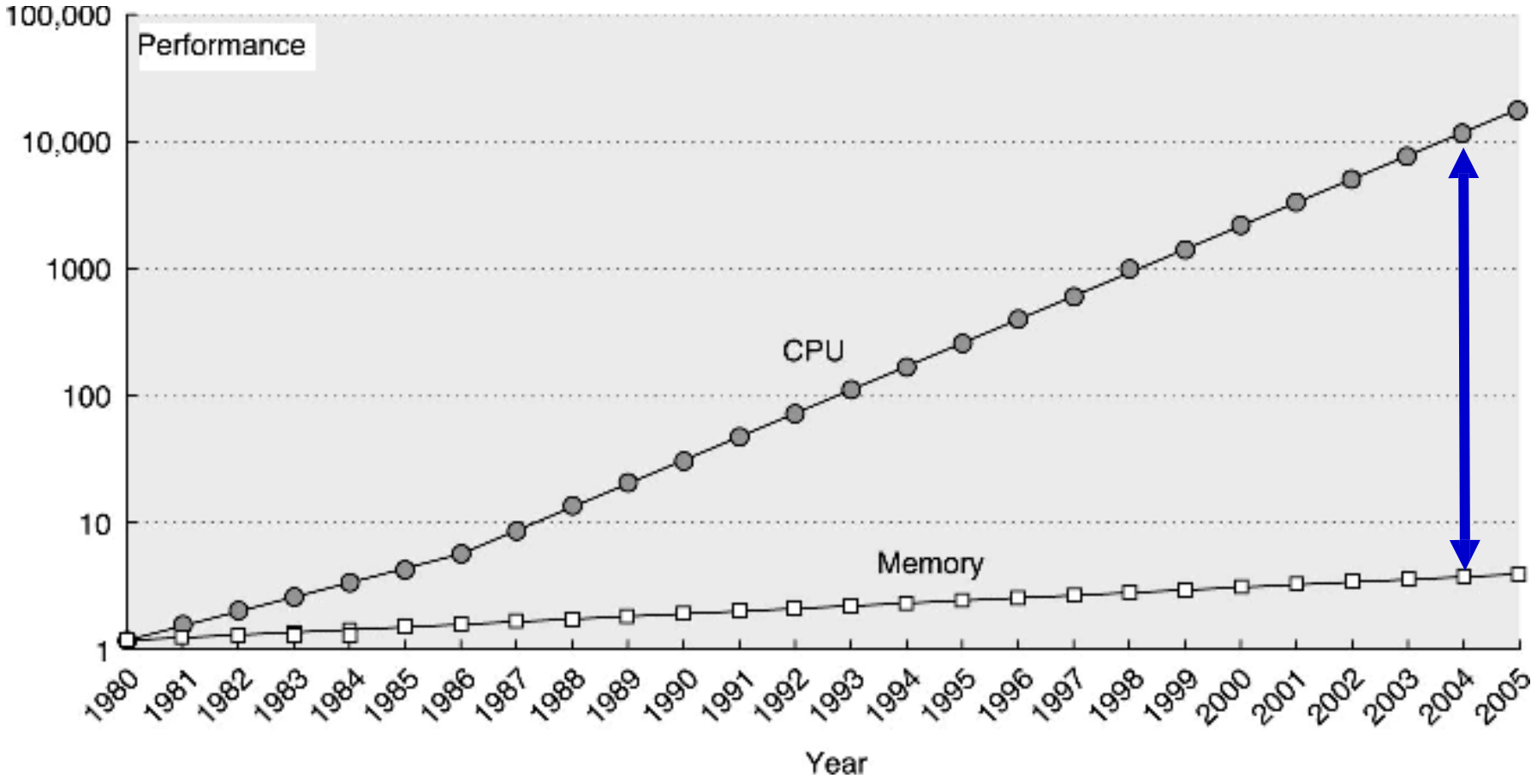
Vitesse des mémoires et des processeurs (1)

- **Évolution**

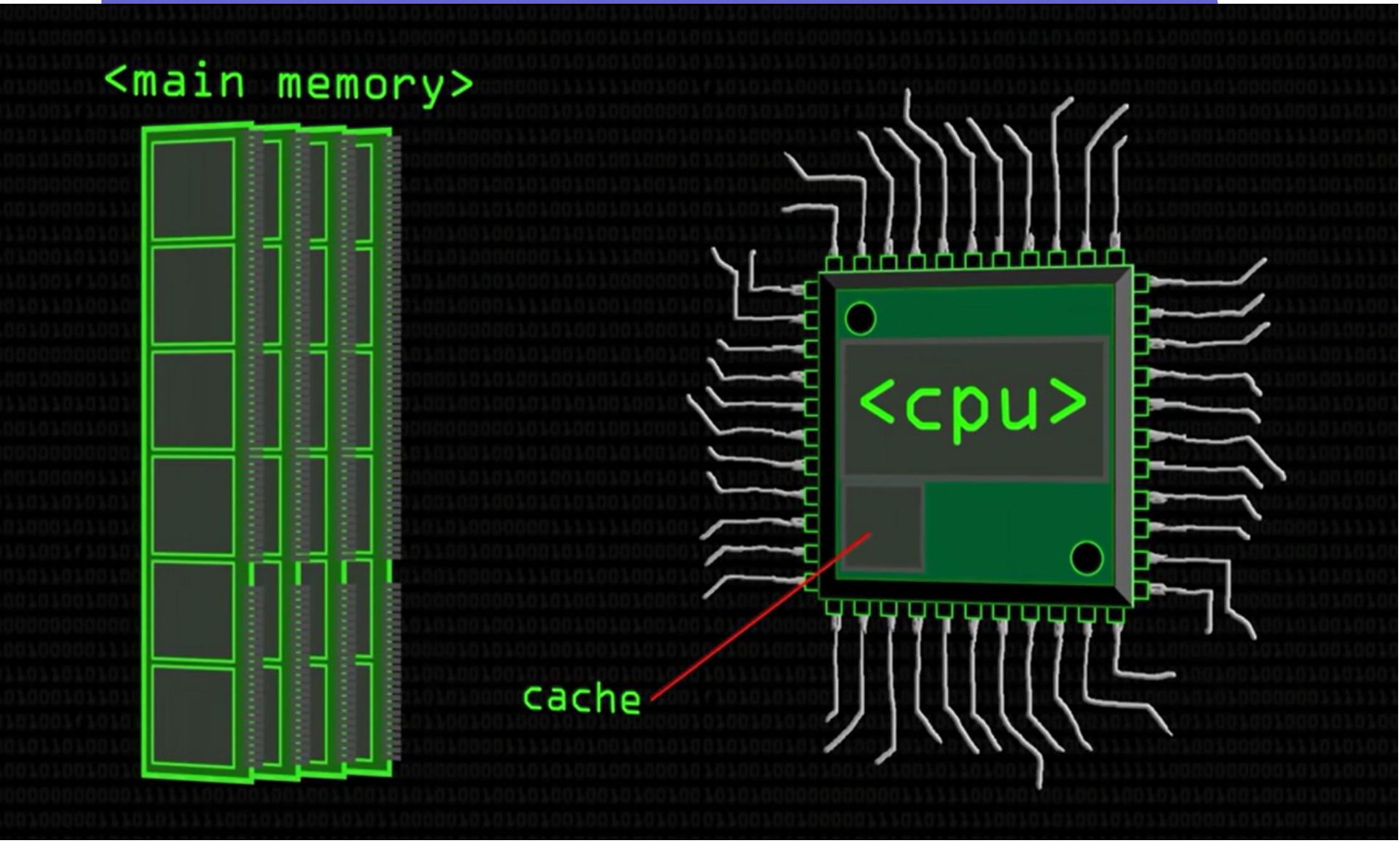
Année	Temps de cycle processeur	Temps de cycle mémoire
1990	~100ns	~140ns
1998	~4ns	~60ns
2002	~0.6ns	~50ns

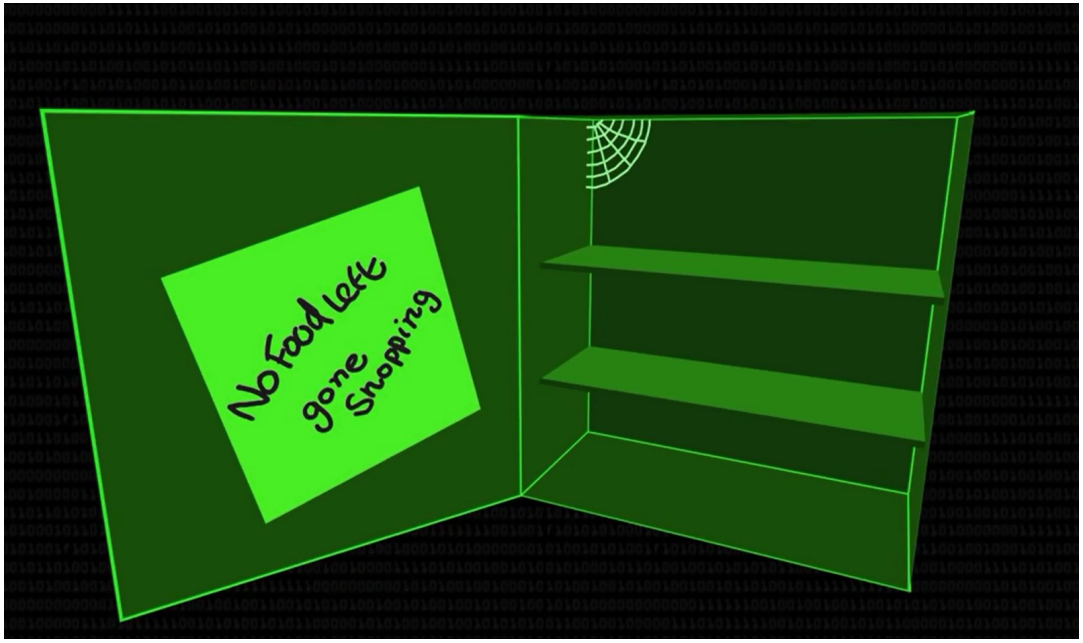
Objectifs et principes du cache

Vitesse des mémoires et des processeurs (2)

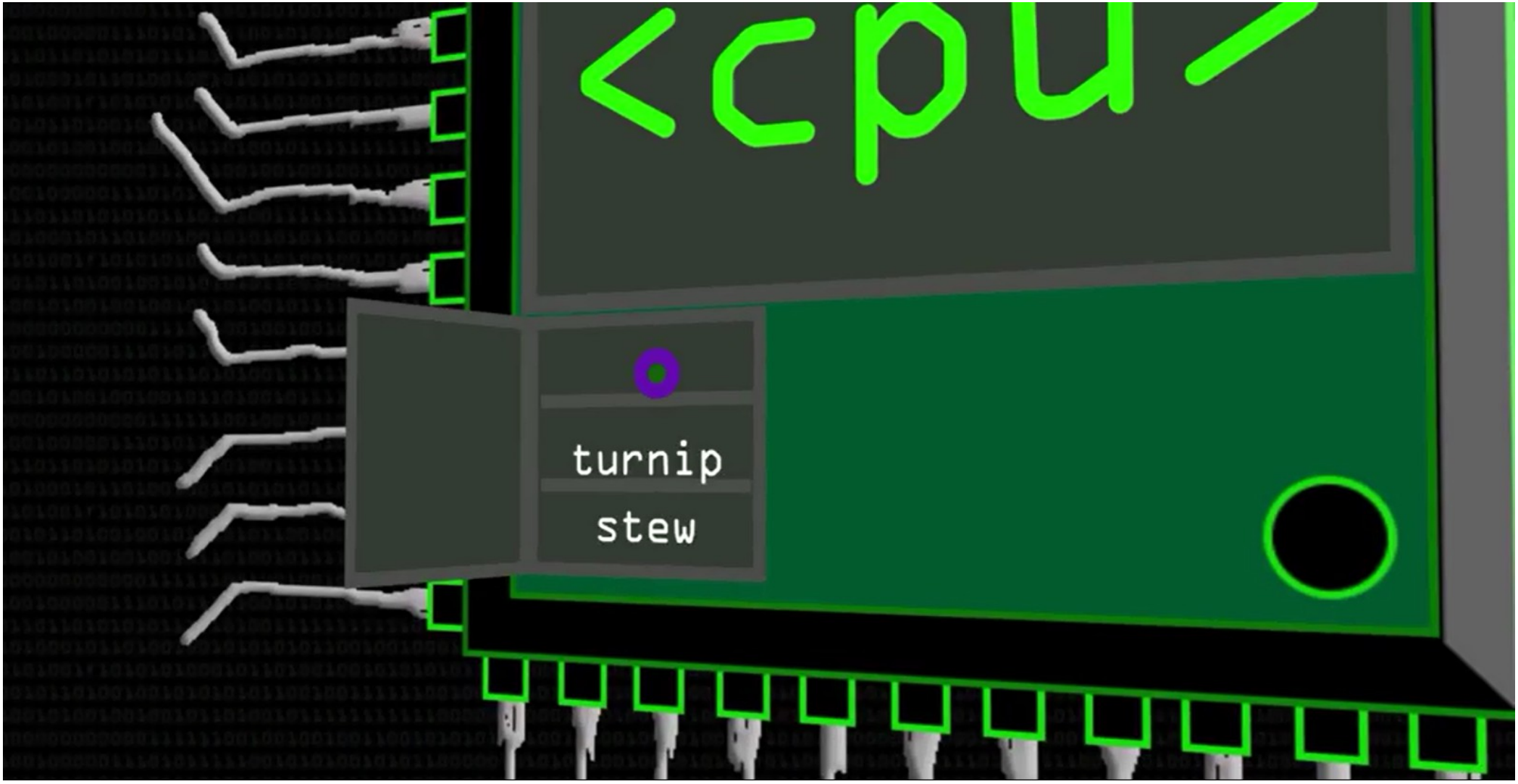


Cache plus rapide




No food left
gone
Shopping

Copie



Objectifs et principes du cache
Principe de localité (1)

- Localité spatiale :
  - Tendance à accéder aux données qui sont proches de celles récemment utilisées

- Localité temporelle :
  - Tendance à réutiliser des données récemment utilisées

Objectifs et principes du cache

Principe de localité (3)

- Les données

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

- y[i]: propriétés de localités temporelle et spatiale.
- a[i][j]: propriété de localité spatiale.
- x[j]: propriété de localité temporelle et spatiale.

- Leprogramme

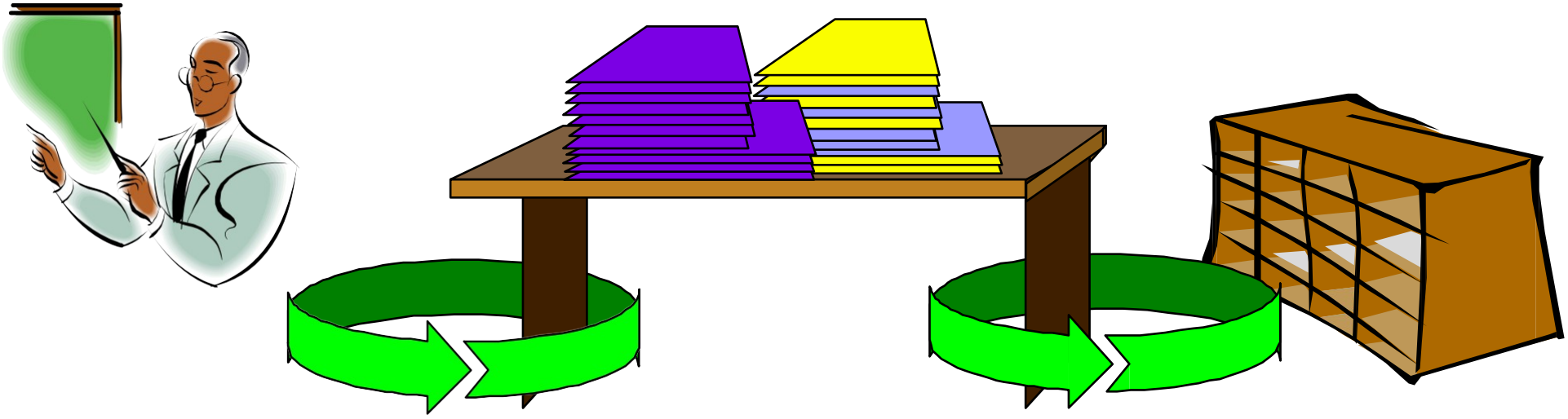
```
                ...
05  LOOP        LDR  R1,  R0,  #3
06               ADD  R1,  R1,  #5
07               STR  R1,  R0,  #30
08               ADD  R0,  R0,  #1
09               ADD  R3,  R0,  R2
0A               BRn  LOOP

                ...
```

Boucle : réutilisation des instructions : localité temporelle

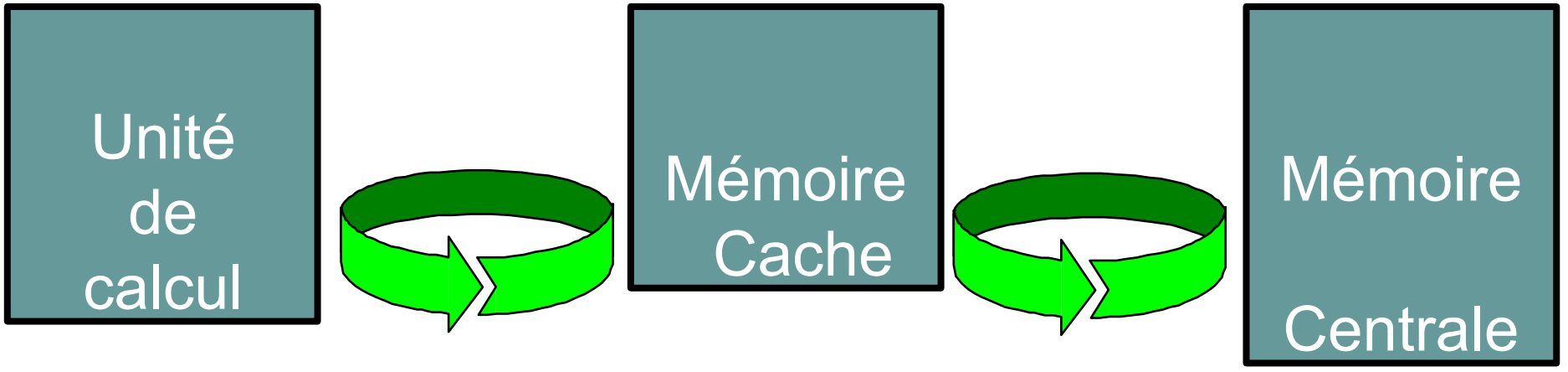
Instructions consécutives en mémoire : localité spatiale

Analogie

- Homme = Unité de calcul
- Le bureau = Mémoire cache
- La bibliothèque = Mémoire centrale

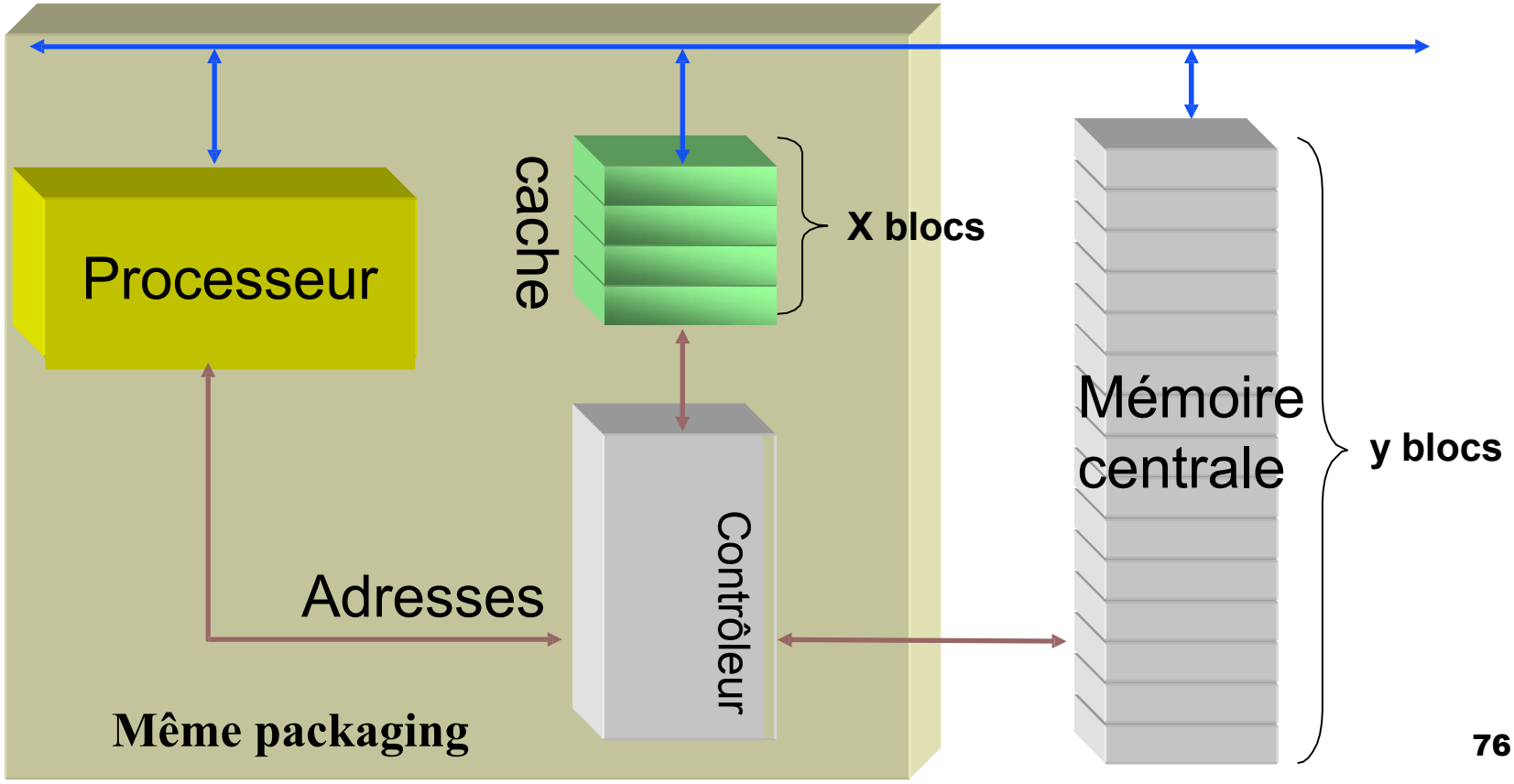


L'élément de base est le bloc
4 octets < Nbre d'octet par blocs < 128 octets)



Organisation mémoire

- La mémoire cache possède x blocs
- La mémoire centrale possède y blocs
- y >> x

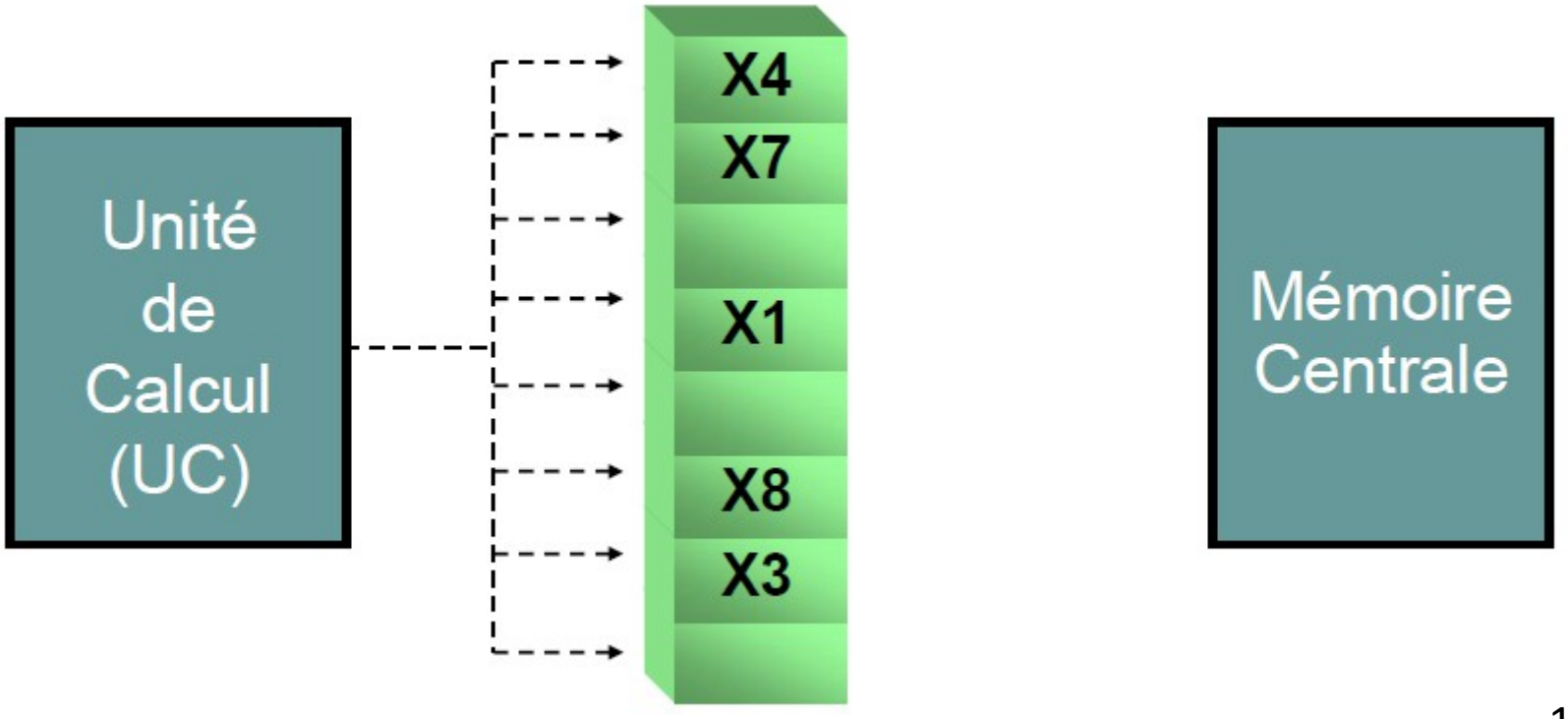


Placement des données dans le cache

- Le placement des données dans le cache est géré par le matériel :
  - le programmeur n'a pas à se soucier du placement des données dans le cache
  - En revanche le programmeur devra prendre en considération la présence du cache pour optimiser les performances.
  - le fonctionnement du cache est transparent pour le programmeur.

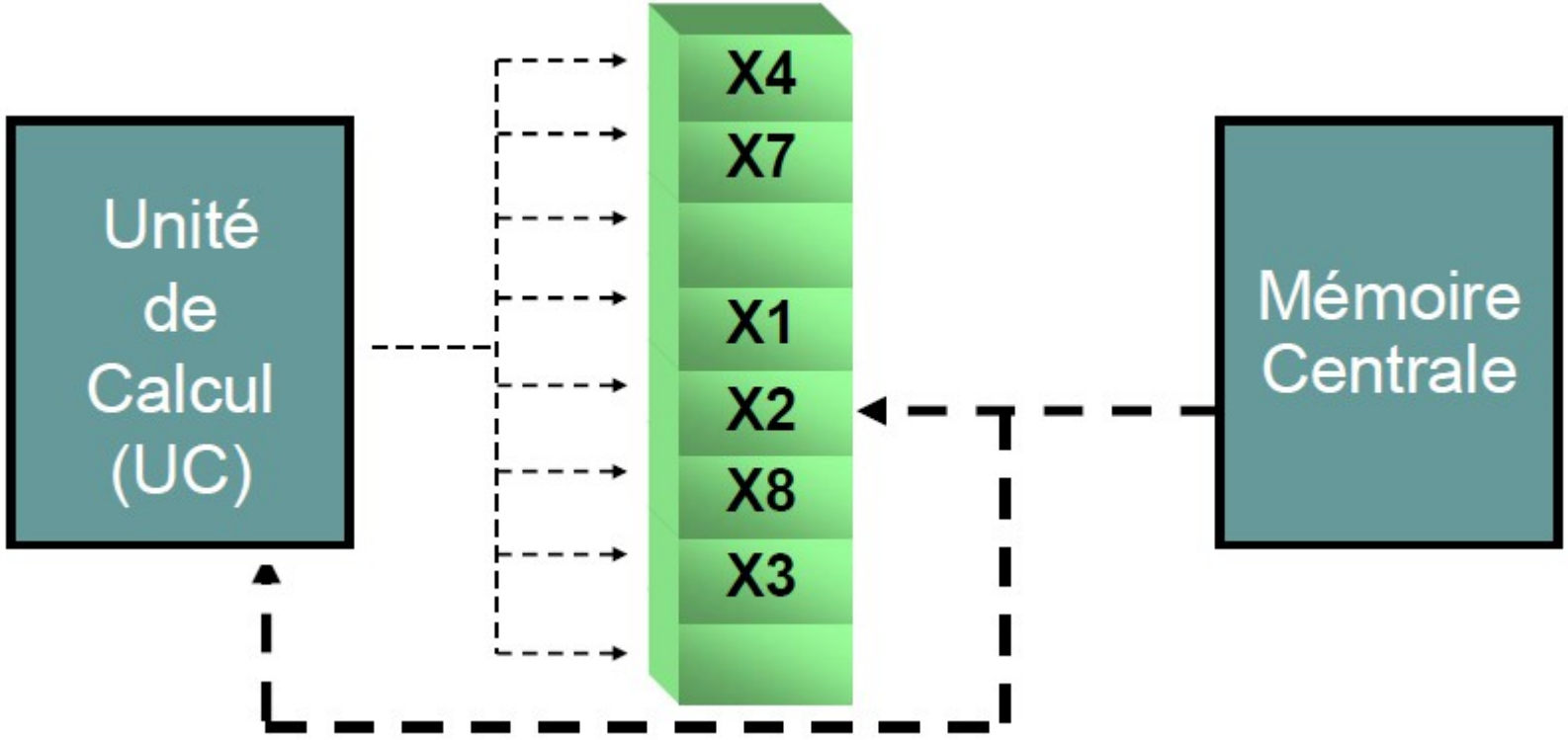
Principe général (1)

- L'UC veut faire référence à un bloc X2 dans le cache
  - Recherche de X2 dans le cache
  - => Défaut de cache



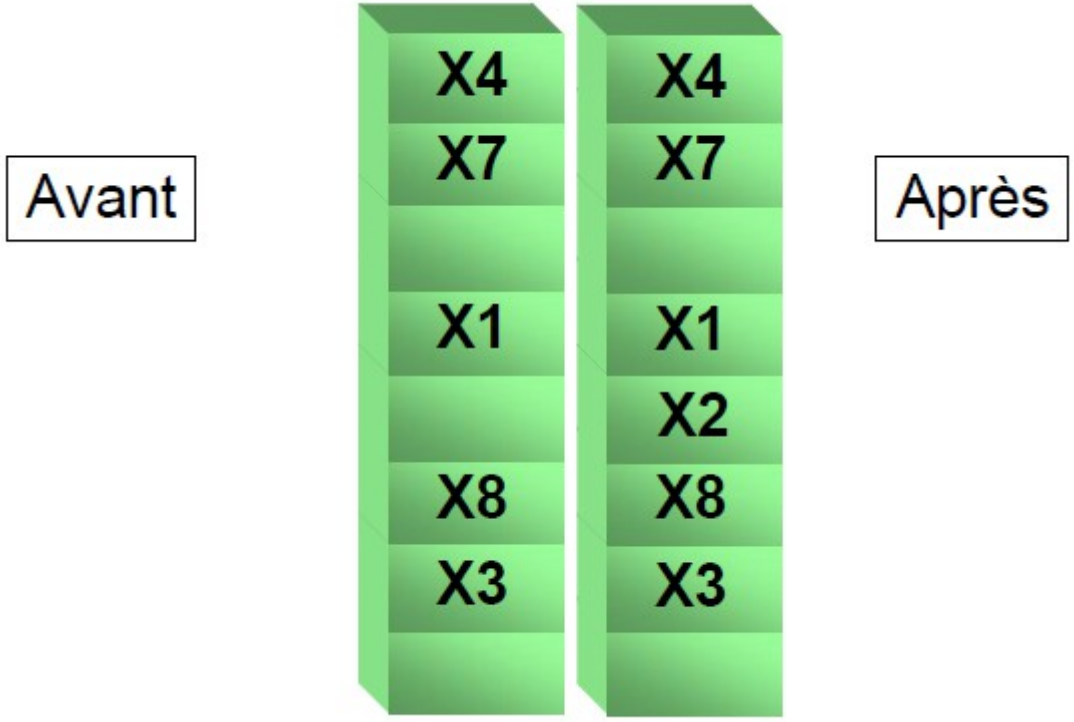
Principe général (2)

- L'UC veut faire référence à un bloc X2 dans le cache
  - Extraction de X2 de la mémoire centrale

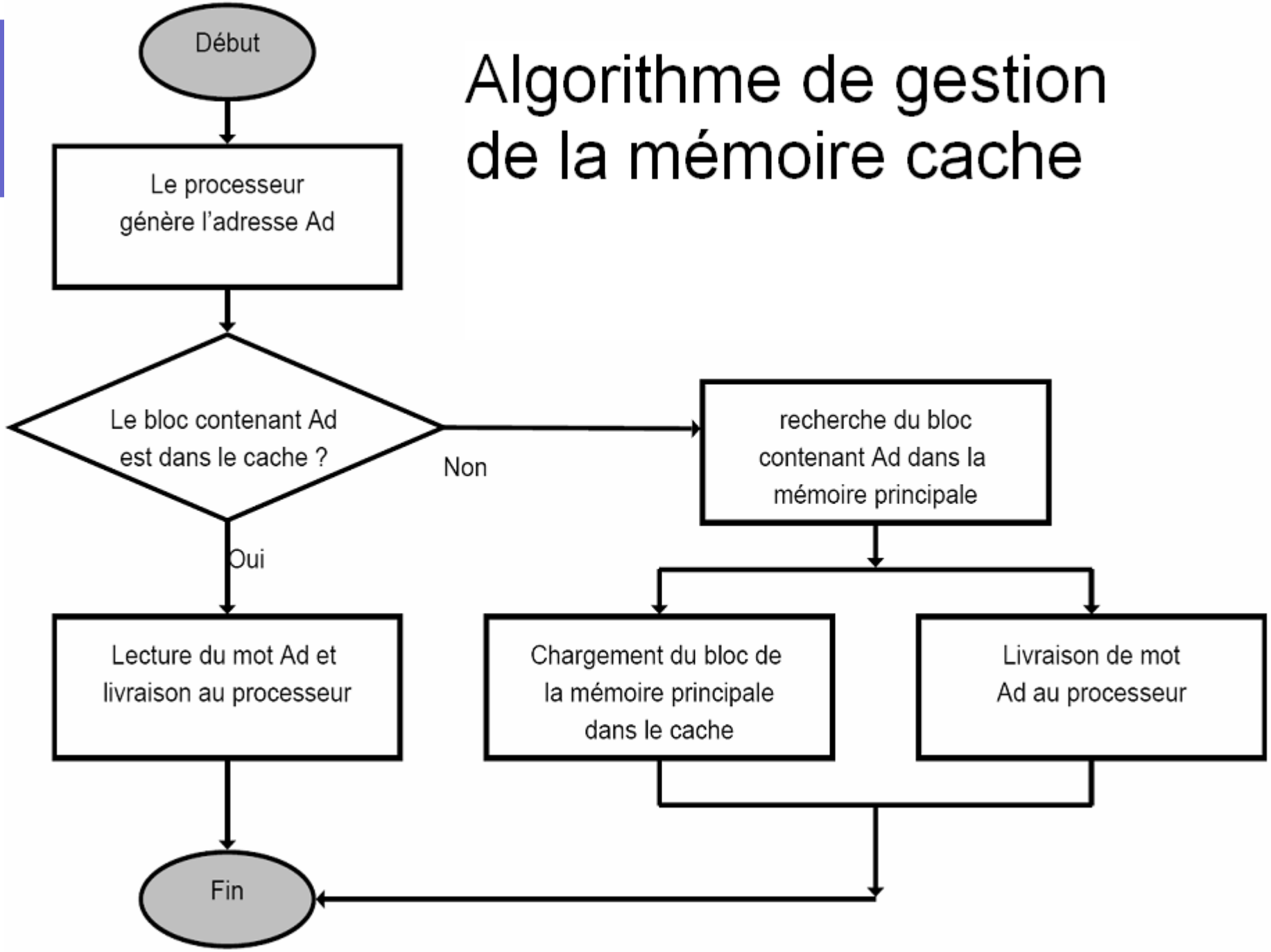


Principe général (3)

- Il y a eu transfert d'un nouveau bloc (X2) de la mémoire centrale, dans la mémoire cache.



Algorithme de gestion de la mémoire cache



Bloc ou ligne de cache (1)

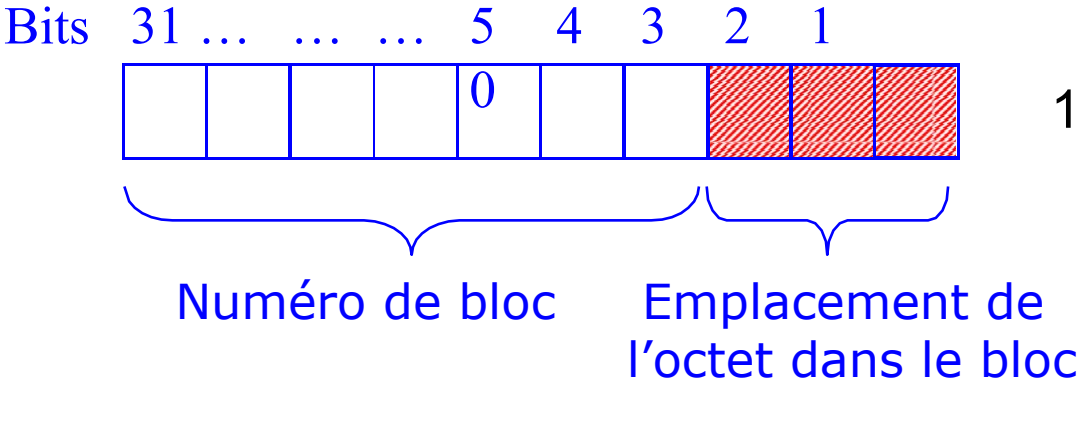
- L'unité d'information qui peut être présente ou non dans le cache est appelée un bloc, qui constitue une ligne (ou rangée) du cache. Les blocs ont généralement entre 4 et 128 octets et sont tous de même taille dans un cache donné.

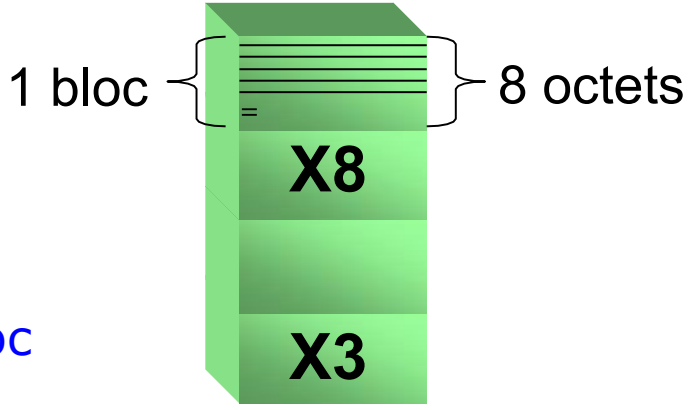
- La mémoire centrale et la mémoire cache ont impérativement les même tailles de blocs.

Bloc ou ligne de cache (2)

- L'adresse fournie par le processeur peut être scindée en deux parties : le n° de bloc et l'adresse dans le bloc.

- Exemple : @ sur 32 bits et blocs de 8 octets

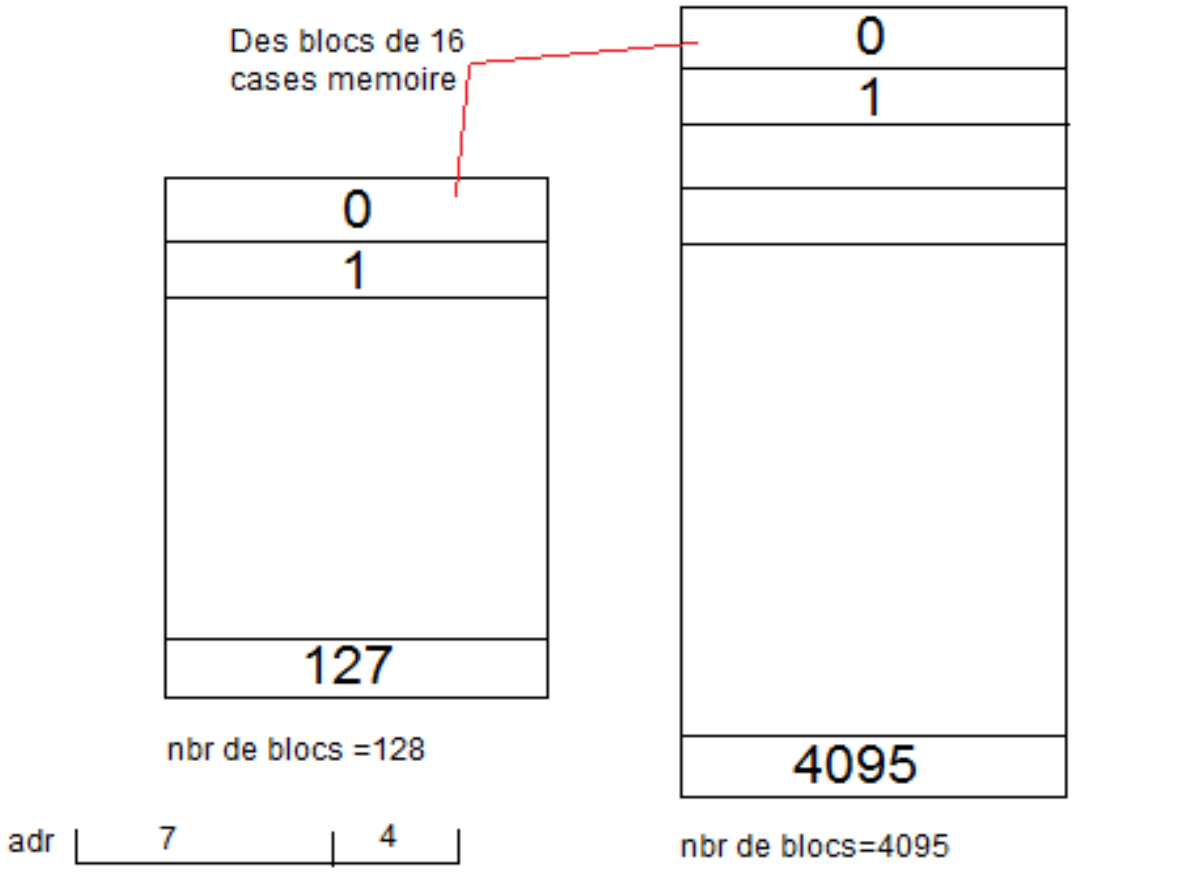




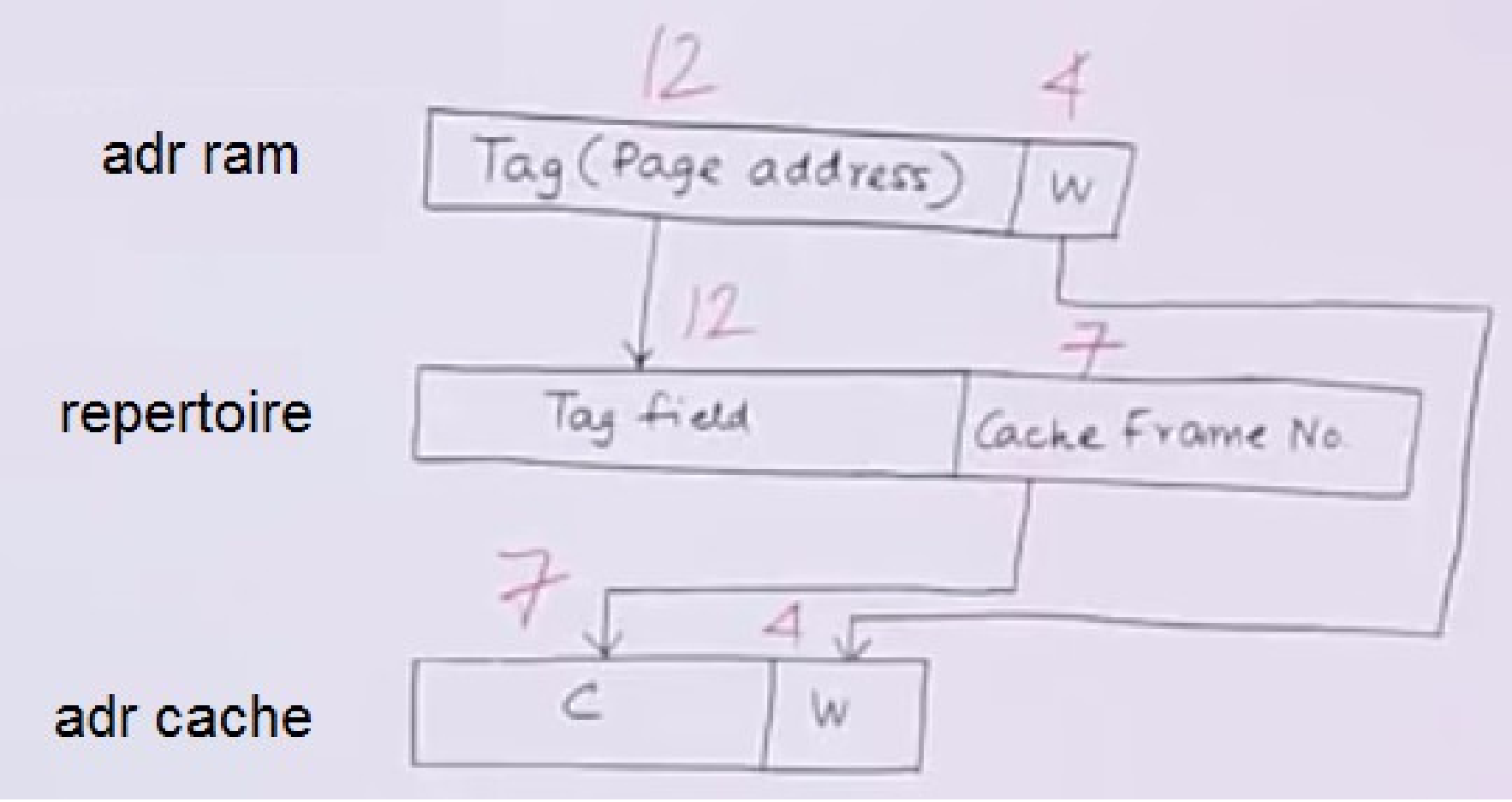
Chapitre 4 : Les mémoires caches

- 4.1 Objectif et principe d'une mémoire cache
- 4.2 Où placer un bloc?
  - **Caches totalement associatifs**
  - **Caches à correspondance directe**
- 4.3 Comment un bloc est-il trouvé?
- 4.4 Quel bloc remplacé lors d'un défaut?
- 4.5 Comment sont traitées les écritures?

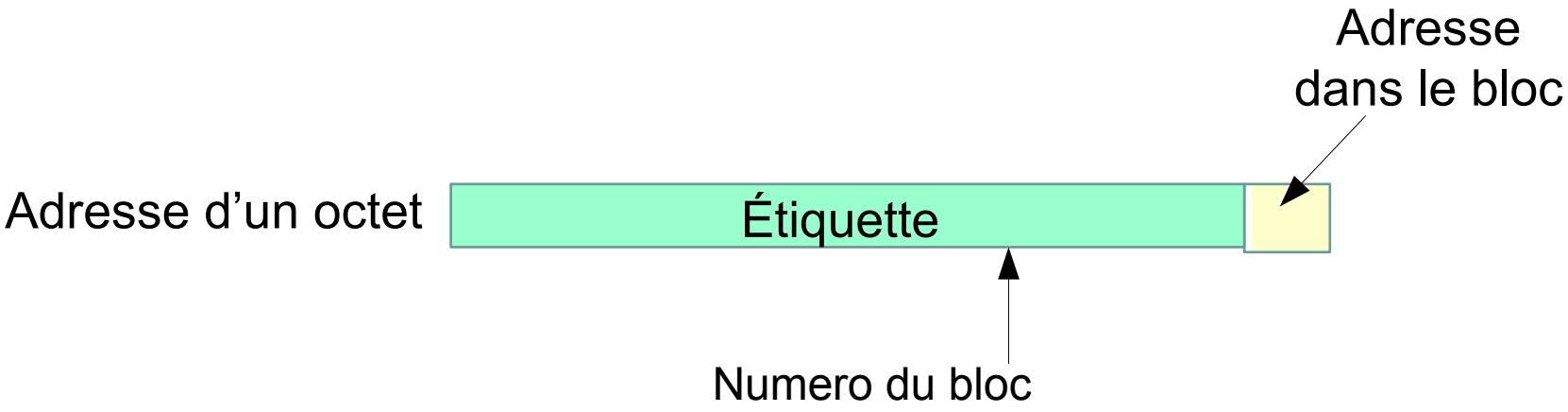
Mapping cache



Mapping associatif

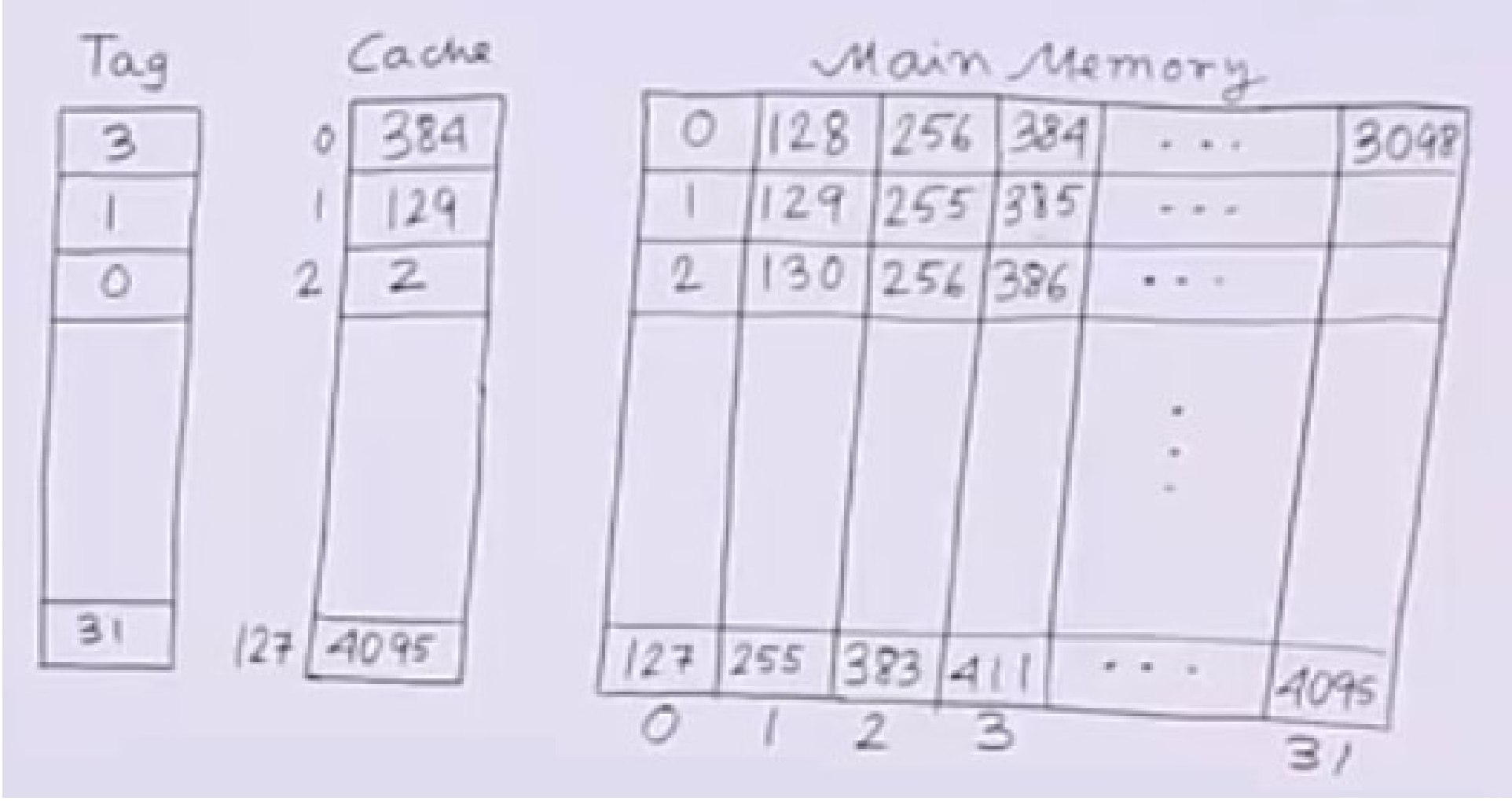


Mapping associatif

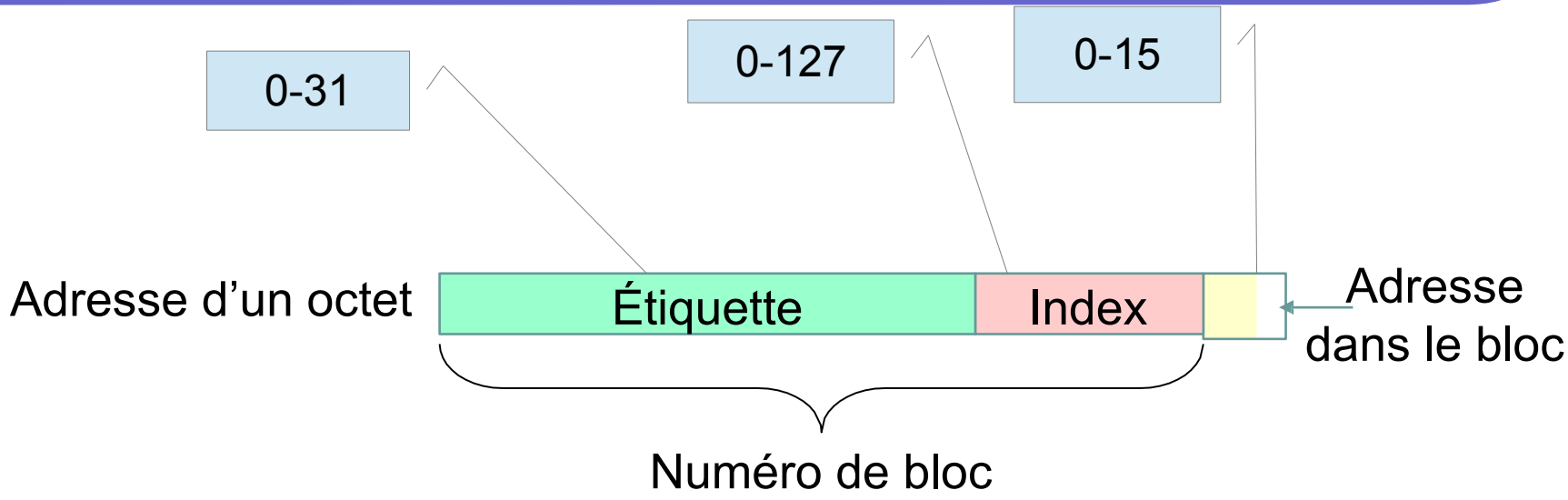


Mapping Direct

4096 / 128 = 32



Cache à correspondance directe



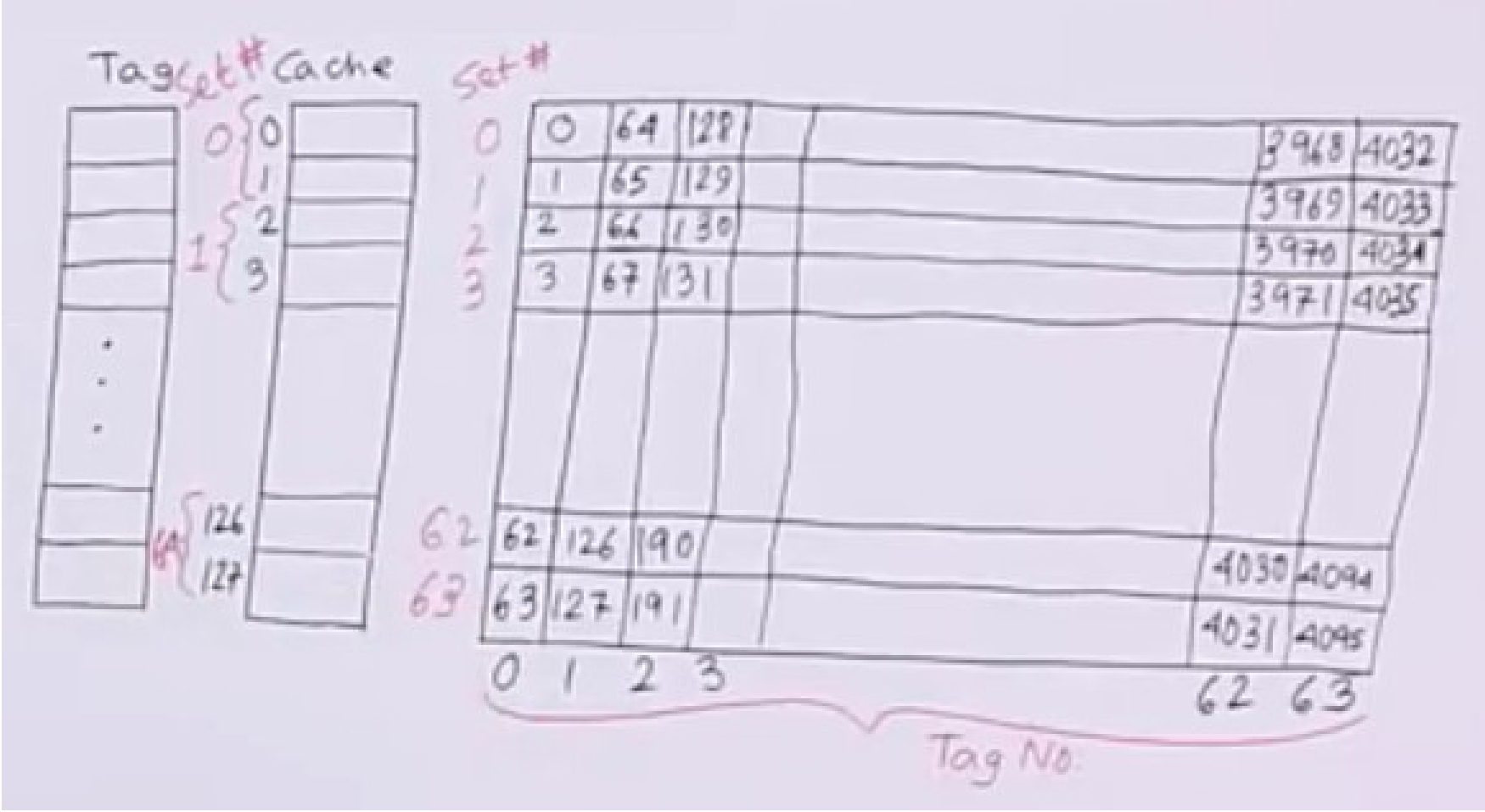
Cache associatif par ensemble(groupe) de bloc

Ensemble = 2 blocs

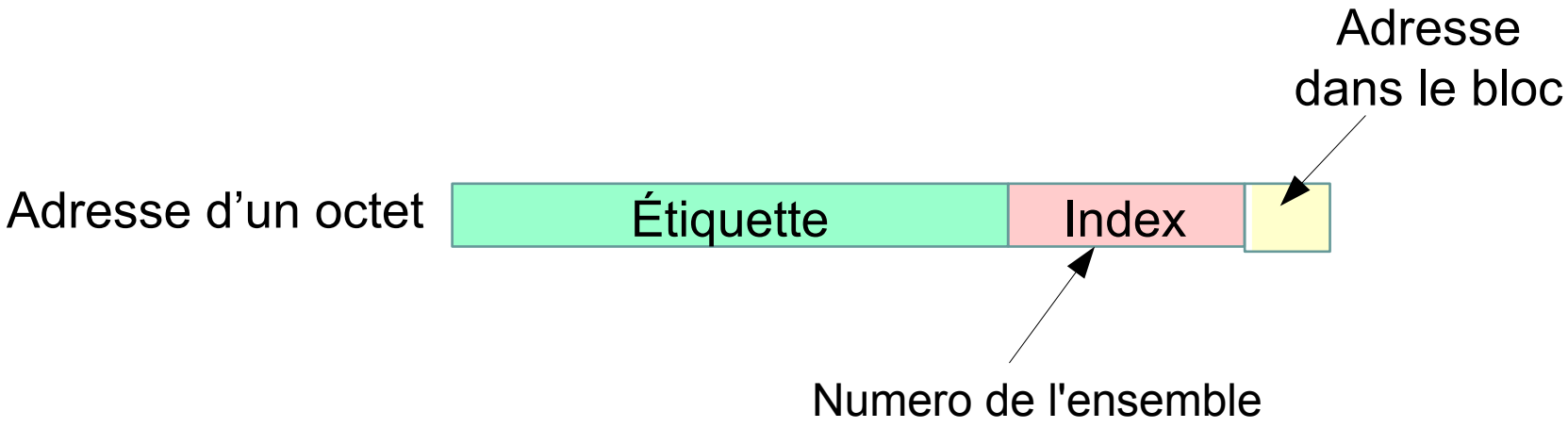
Cache : 128 blocs et 128 / 2 = 64 ensembles
64 = 2^6

RAM : 4096 blocs et 4096 / 64 = 64 enssembles
64=2^6

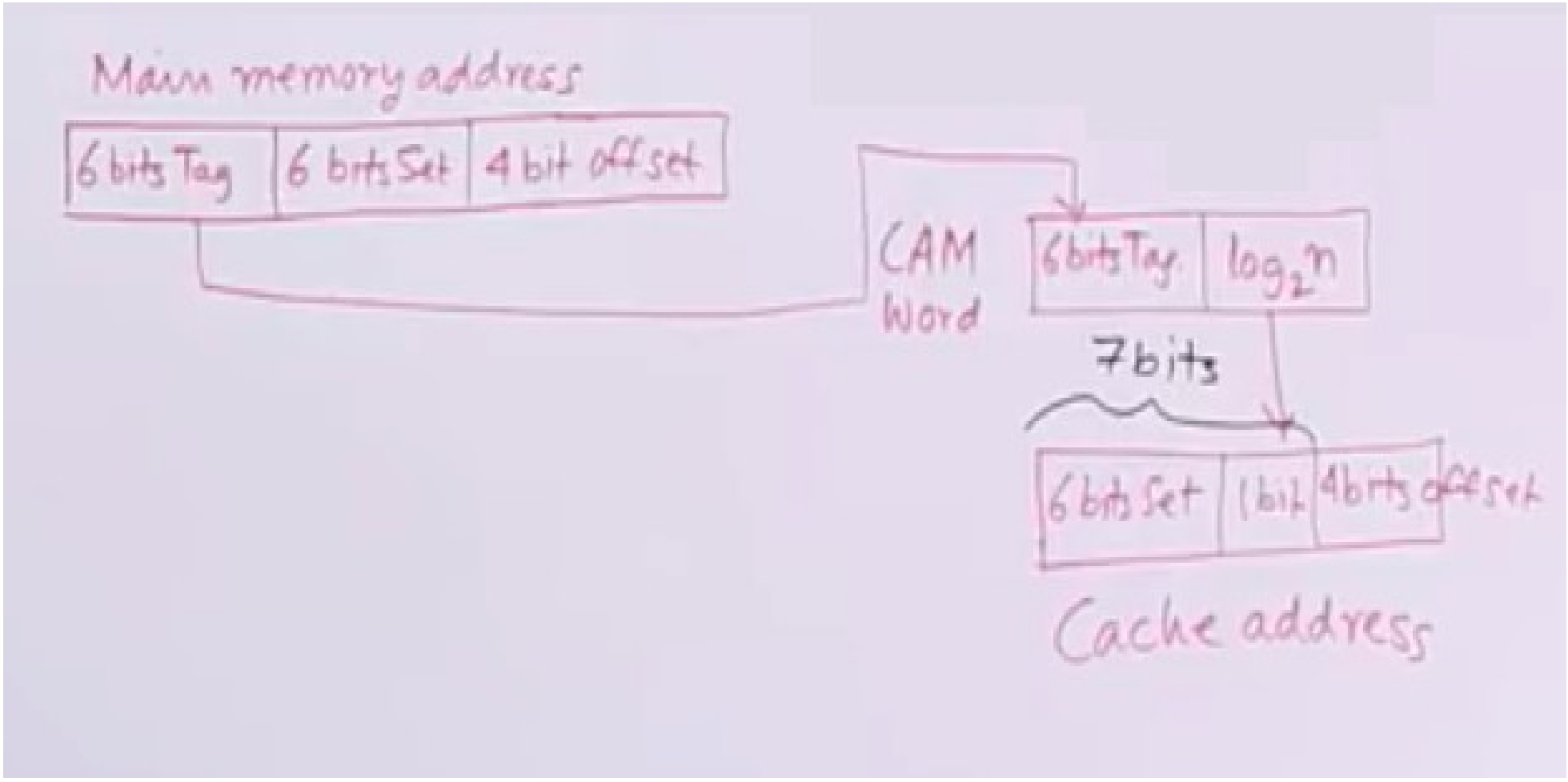
Cache associatif par ensemble(groupe) de bloc



Cache associatif par ensemble(groupe) de bloc



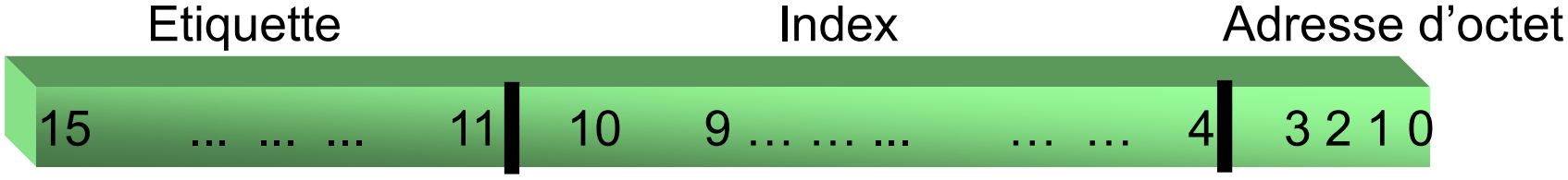
Cache associatif par ensemble(groupe) de bloc



Les étiquettes en fonction du type de caches

- Cache associatif

- Cache à correspondance directe



- Cache associatif par ensemble de bloc

Chapitre 4 : Les mémoires caches

- 4.1 Objectif et principe d'une mémoire cache
- 4.2 Où placer un bloc?
- 4.3 Comment un bloc est-il trouvé?
  - **Caches totalement associatifs**
  - **Caches à correspondance directe**
- 4.4 Quel bloc remplacé lors d'un défaut?
- 4.5 Comment sont traitées les écritures?

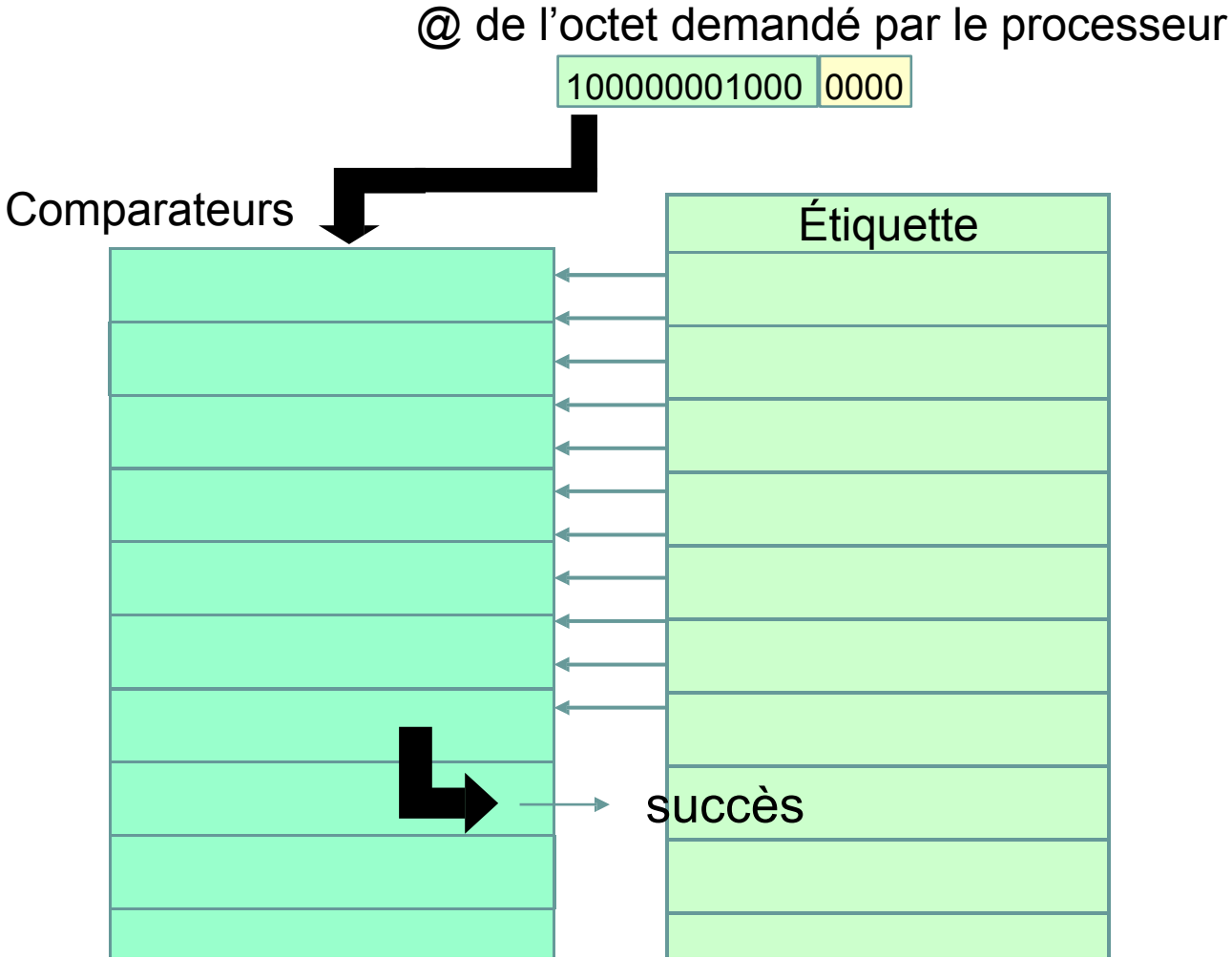
Comment un bloc est-il trouvé?
Recherche d'un bloc dans le cache

- Le processeur demande une adresse memoire (RAM)
- L'adresse est analysée pour savoir si la données est dans le cache et pour connaître le numero du bloque.
- Si trouvée => Succès Sinon => Défaut

Comment un bloc est-il trouvé?

Cas du Cache associatif

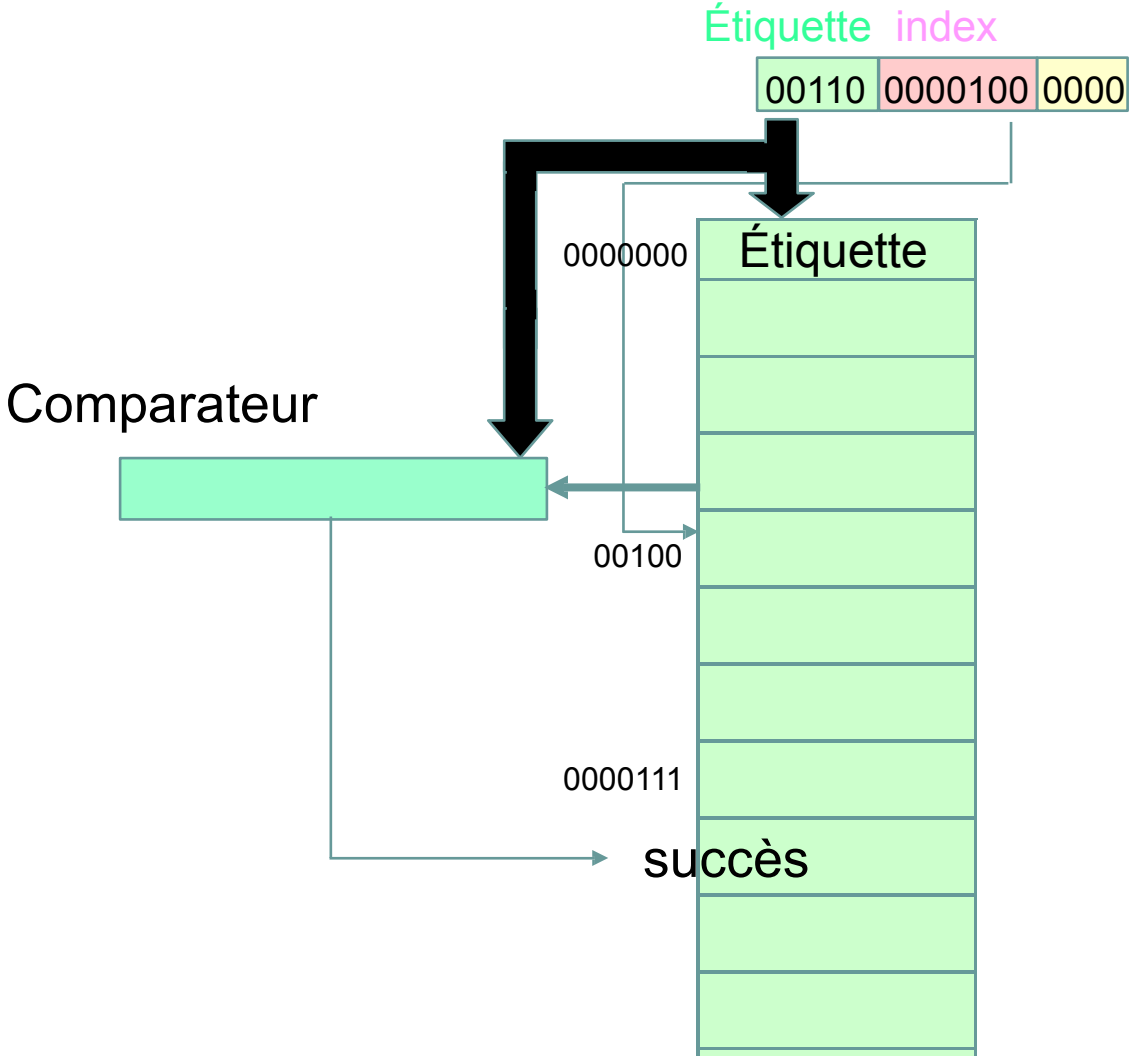
- Comparaison entre l'étiquette de l'adresse et celle rangé en mémoire cache.



Comment un bloc est-il trouvé?

Cas du Cache à correspondance directe

- Comparaison entre l'étiquette et celle de l'adresse rangée en mémoire cache.



Les mémoires caches

- 1 Objectif et principe d'une mémoire cache
- 2 Où placer un bloc?
- 3 Comment un bloc est-il trouvé?
- 4 Quel bloc remplacé lors d'un défaut?
- 5 Comment sont traitées les écritures?

Quel bloc remplacé lors d'un défaut ?

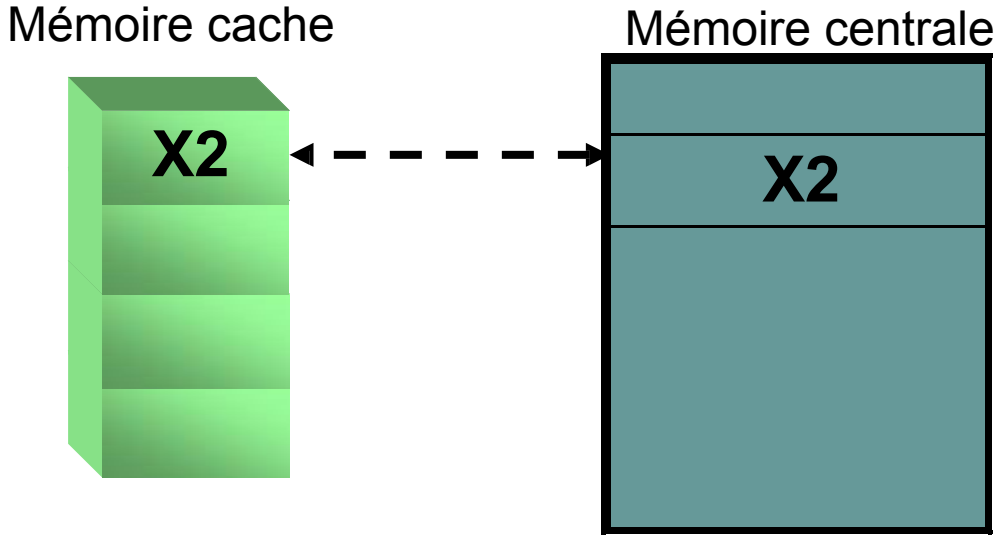
- Remplacement aléatoire :
  - Simplicité de l'algorithme
- FIFO (First In First Out)
  - Simplicité de conception
- LRU (Least Recently Used)
  - Circuits complexes.

Chapitre 4 : Les mémoires caches

- 4.1 Objectif et principe d'une mémoire cache
- 4.2 Où placer un bloc?
- 4.3 Comment un bloc est-il trouvé?
- 4.4 Quel bloc remplacé lors d'un défaut?
- 4.5 Comment sont traitées les écritures?

Gestion des écritures (1)

- Quand une donnée se situe dans le cache, le système en possède deux copies :
  - une dans la mémoire principale
  - une dans la mémoire cache
- Comment gérer les mises à jour lorsque la donnée est modifiée localement?



Gestion des écritures (2)

- *write-through* : La donnée est écrite à la fois dans le cache et dans la mémoire principale. La mémoire principale et le cache ont à tout moment une valeur identique
- *write-back* : L'information n'est écrite dans la mémoire principale que lorsque la ligne disparaît du cache. Cette technique est la plus répandue car elle permet d'éviter de nombreuses écritures mémoires inutiles. Pour ne pas avoir à écrire des informations qui n'ont pas été modifiées (et ainsi éviter d'encombrer inutilement le bus), chaque ligne de la mémoire cache est pourvue d'un bit *dirty*. Lorsque la ligne est modifiée dans le cache, ce bit est positionné à 1, indiquant qu'il faudra réécrire la donnée dans la mémoire principale.

Plusieurs niveaux de cache