

# Mémoire 2/2

Thomas Lavergne  
lavergne@lisn.fr

Mémoire virtuelle

## Problème

- Plusieurs centaines de processus sur le système
- Certains sont très gros (plusieurs Go)

Somme des tailles des processus  $>$  Capacité RAM

## Problème

- Plusieurs centaines de processus sur le système
- Certains sont très gros (plusieurs Go)

Somme des tailles des processus  $>$  Capacité RAM

## Données inutilisées

- Code de gestion d'erreurs
- Gros tableau : pas tout en même temps
- Bibliothèque : très variable

Ne charger que ce qui est utile !

## Mémoire virtuelle

Chaque processus peut adresser plus d'espace qu'il n'y a effectivement de mémoire physique.

## Mémoire virtuelle

Chaque processus peut adresser plus d'espace qu'il n'y a effectivement de mémoire physique.

## Avantages

- Plus de processus en parallèle
- Moins de soucis de gestion de la mémoire

## Mémoire virtuelle

Chaque processus peut adresser plus d'espace qu'il n'y a effectivement de mémoire physique.

## Avantages

- Plus de processus en parallèle
- Moins de soucis de gestion de la mémoire

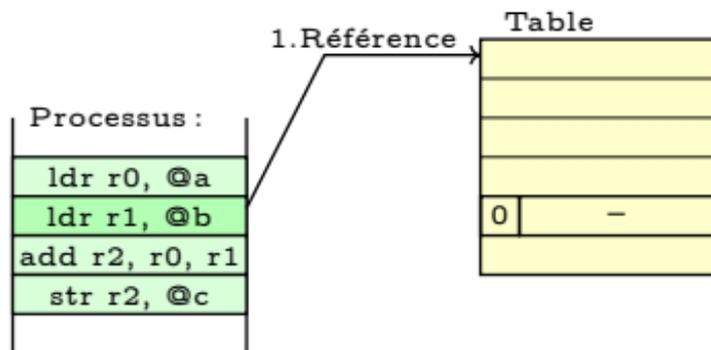
## Pagination à la demande

- Table des pages :  $V = 0$  page sur le disque
- L'OS charge les pages manquantes si nécessaire

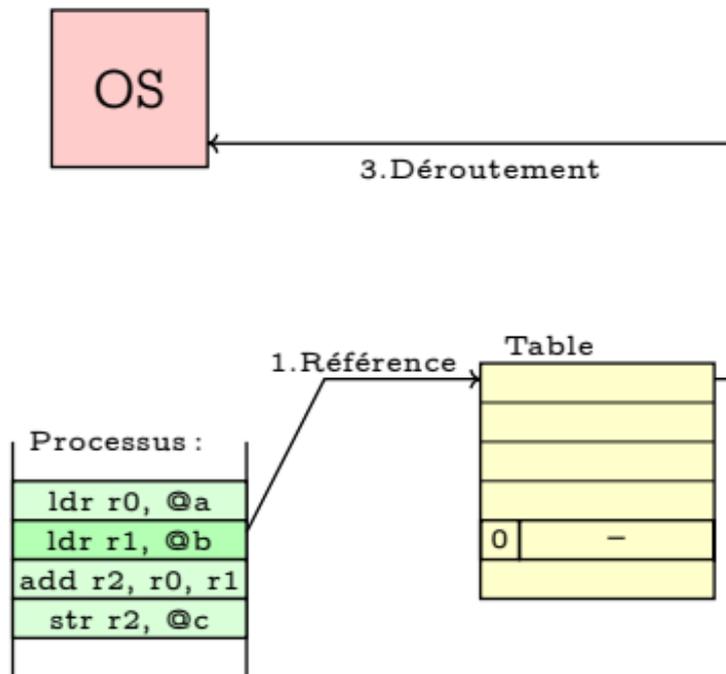
Processus :

ldr r0, @a
ldr r1, @b
add r2, r0, r1
str r2, @c

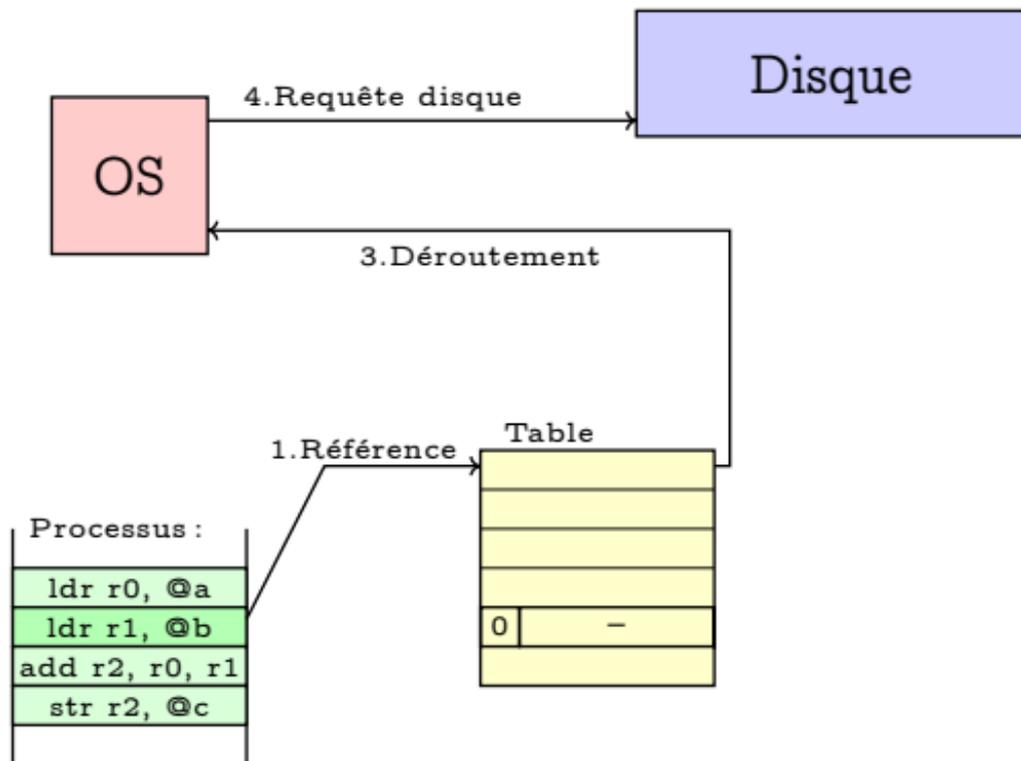
# Pagination à la demande



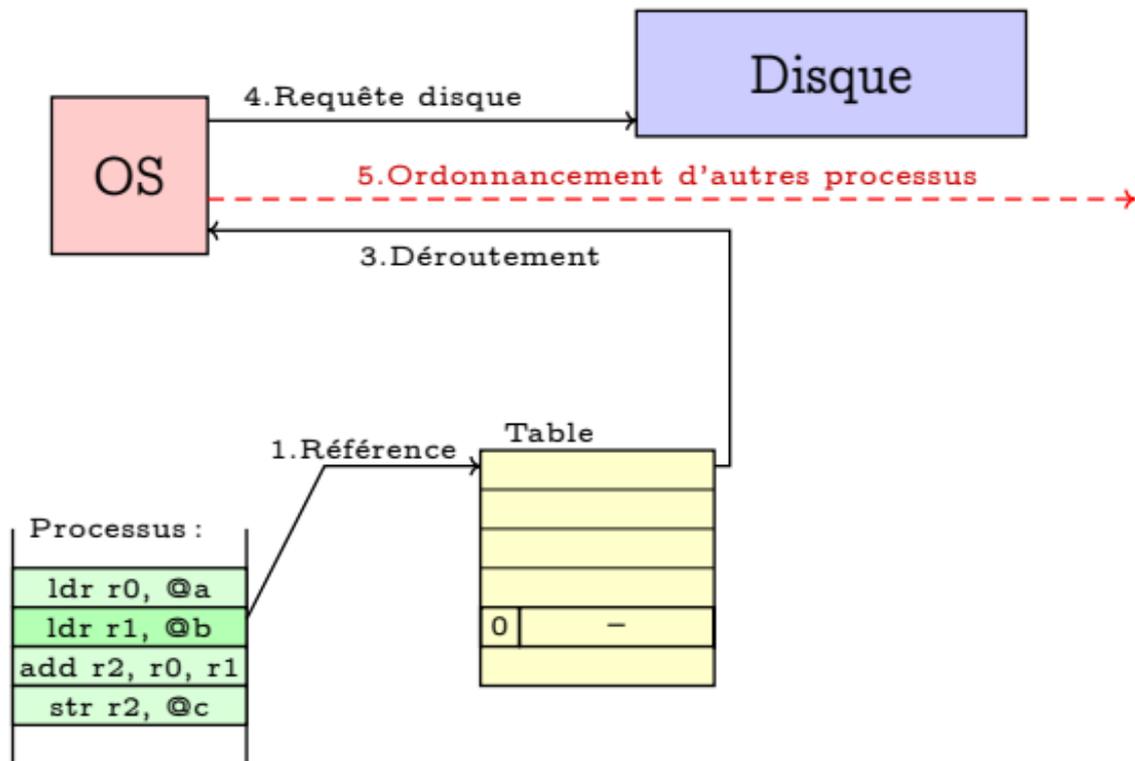
# Pagination à la demande



# Pagination à la demande

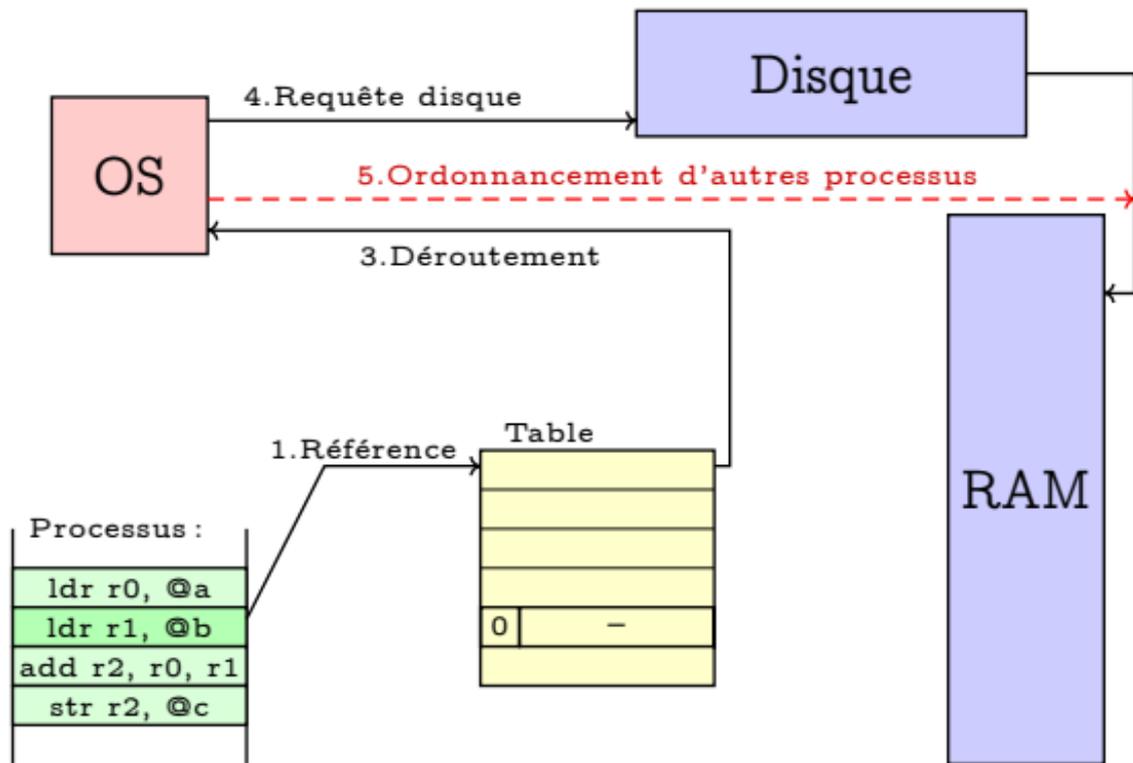


# Pagination à la demande



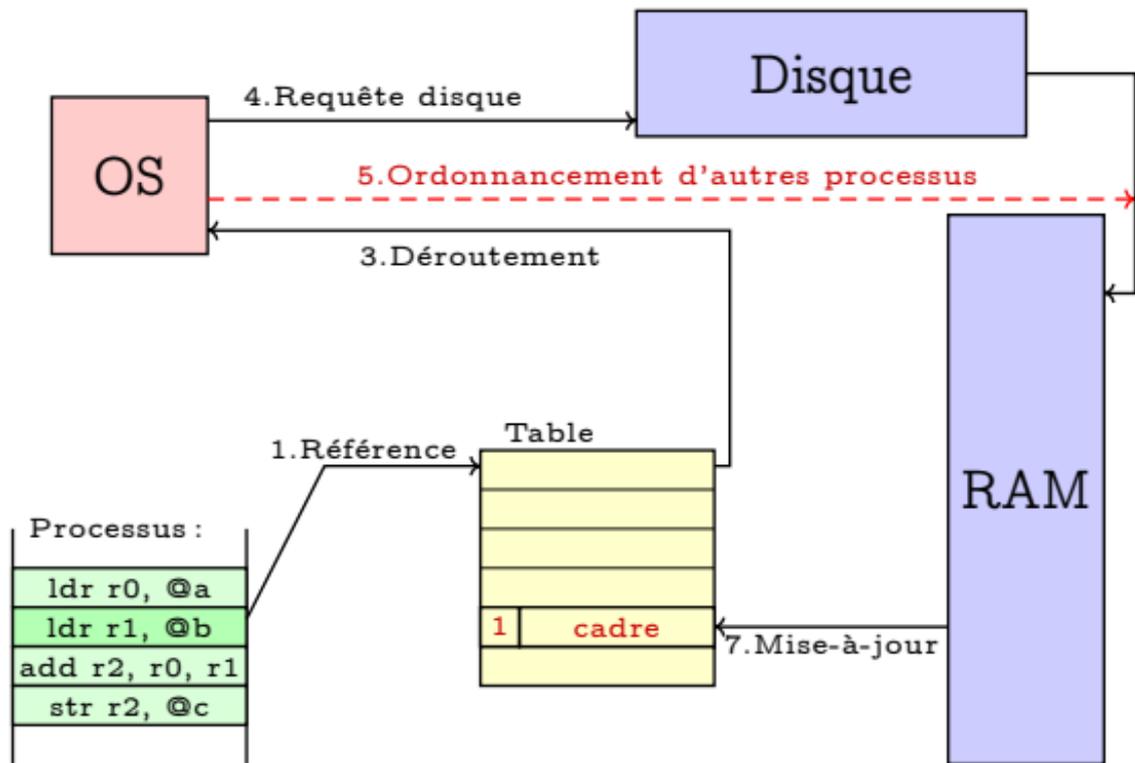
# Pagination à la demande

6. Chargement de la page



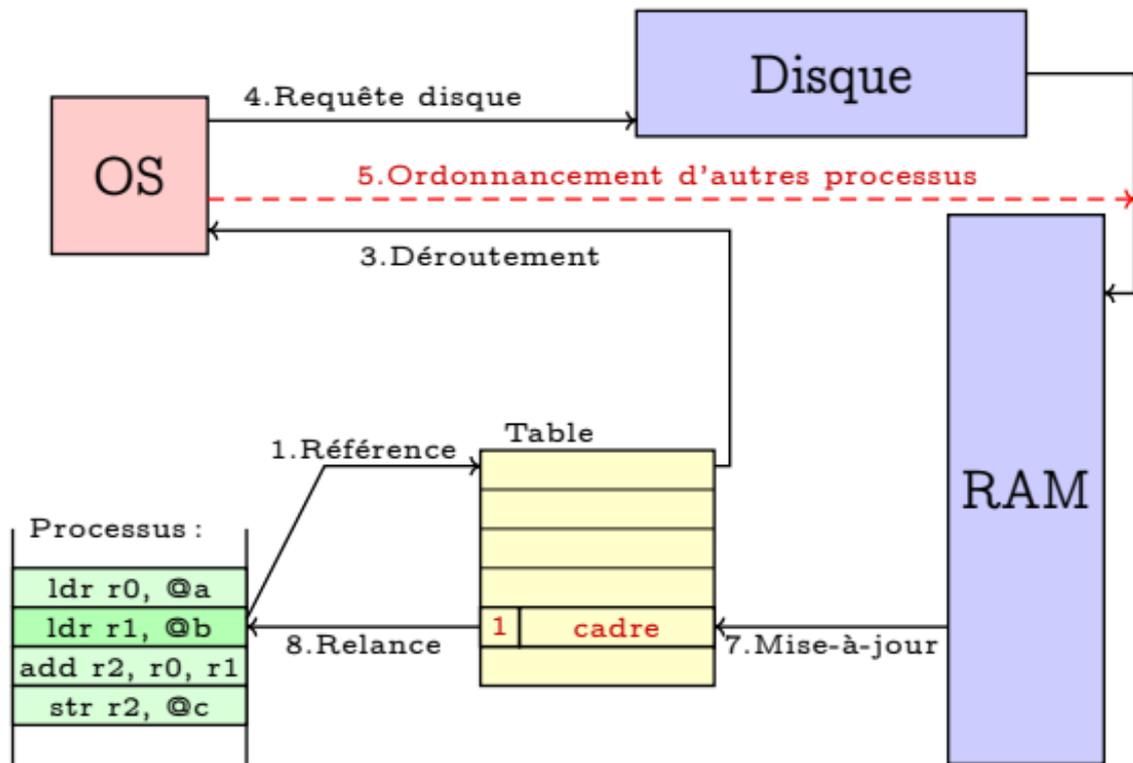
# Pagination à la demande

6. Chargement de la page



# Pagination à la demande

6.Chargement de la page



## Coût des défauts de pages

- $p$  = probabilité de défaut de page
- $M$  = temps d'accès à la mémoire
- $D$  = temps de traitement du défaut

$$\text{Temps d'accès} = (1-p) \times M + p \times D = M + p(D - M)$$

## Coût des défauts de pages

- $p$  = probabilité de défaut de page
- $M$  = temps d'accès à la mémoire
- $D$  = temps de traitement du défaut

$$\text{Temps d'accès} = (1-p) \times M + p \times D = M + p(D - M)$$

## Propriété

Le temps d'accès à la mémoire est, en moyenne, proportionnel à la probabilité de défaut de page.

## Avantages

- ✓ (presque) plus de fragmentation
- ✓ implémentation simple
- ✓ indépendante du processus

## Avantages

- ✓ (presque) plus de fragmentation
- ✓ implémentation simple
- ✓ indépendante du processus

## Inconvénients

- ✗ taille de page arbitraire
- ✗ découpage non sémantique du processus

Taille de page arbitraire

Impossible de vérifier finement les accès en dehors de la mémoire

### Taille de page arbitraire

Impossible de vérifier finement les accès en dehors de la mémoire

### Découpage non sémantique

Difficile de correctement gérer les permissions sans gaspiller de mémoire

**X** pas de mélange code/données dans une page

### Taille de page arbitraire

Impossible de vérifier finement les accès en dehors de la mémoire

### Découpage non sémantique

Difficile de correctement gérer les permissions sans gaspiller de mémoire

**X** pas de mélange code/données dans une page

### Solution

Paginer un espace segmenté...

# Pagination avec segmentation

### 1er niveau

L'espace de mémoire logique est géré de manière segmentée.

- la traduction d'adresse permet d'obtenir une **adresse linéaire**

## 1er niveau

L'espace de mémoire logique est géré de manière segmentée.

- la traduction d'adresse permet d'obtenir une **adresse linéaire**

## 2ème niveau

L'espace linéaire est géré de manière paginée.

- la traduction d'adresse permet d'obtenir une **adresse physique**

## 1er niveau

L'espace de mémoire logique est géré de manière segmentée.

- la traduction d'adresse permet d'obtenir une **adresse linéaire**

## 2ème niveau

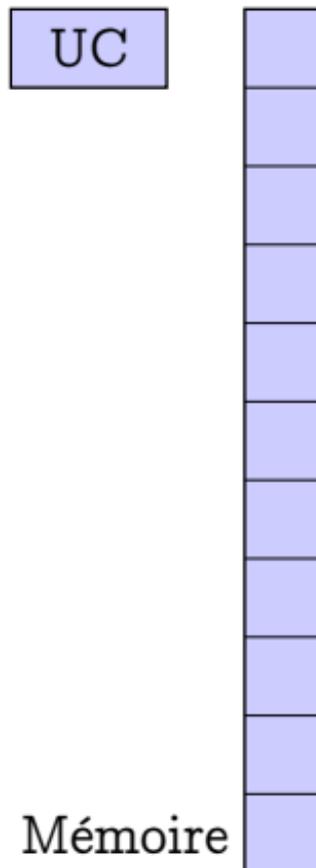
L'espace linéaire est géré de manière paginée.

- la traduction d'adresse permet d'obtenir une **adresse physique**

Le meilleur des deux mondes

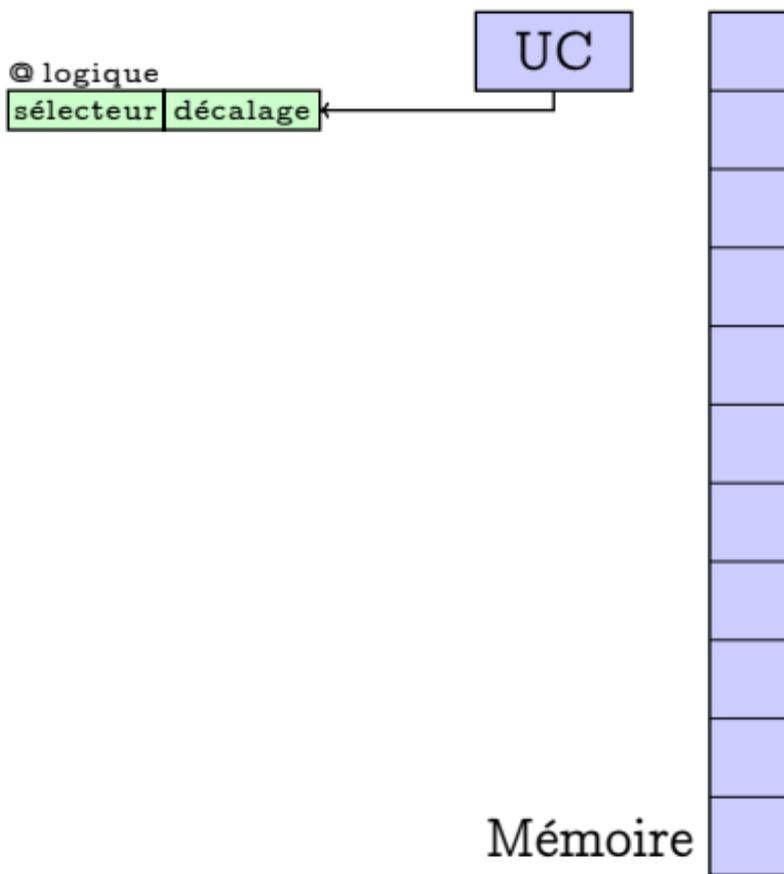
# Segmentation et pagination à deux niveaux

UC

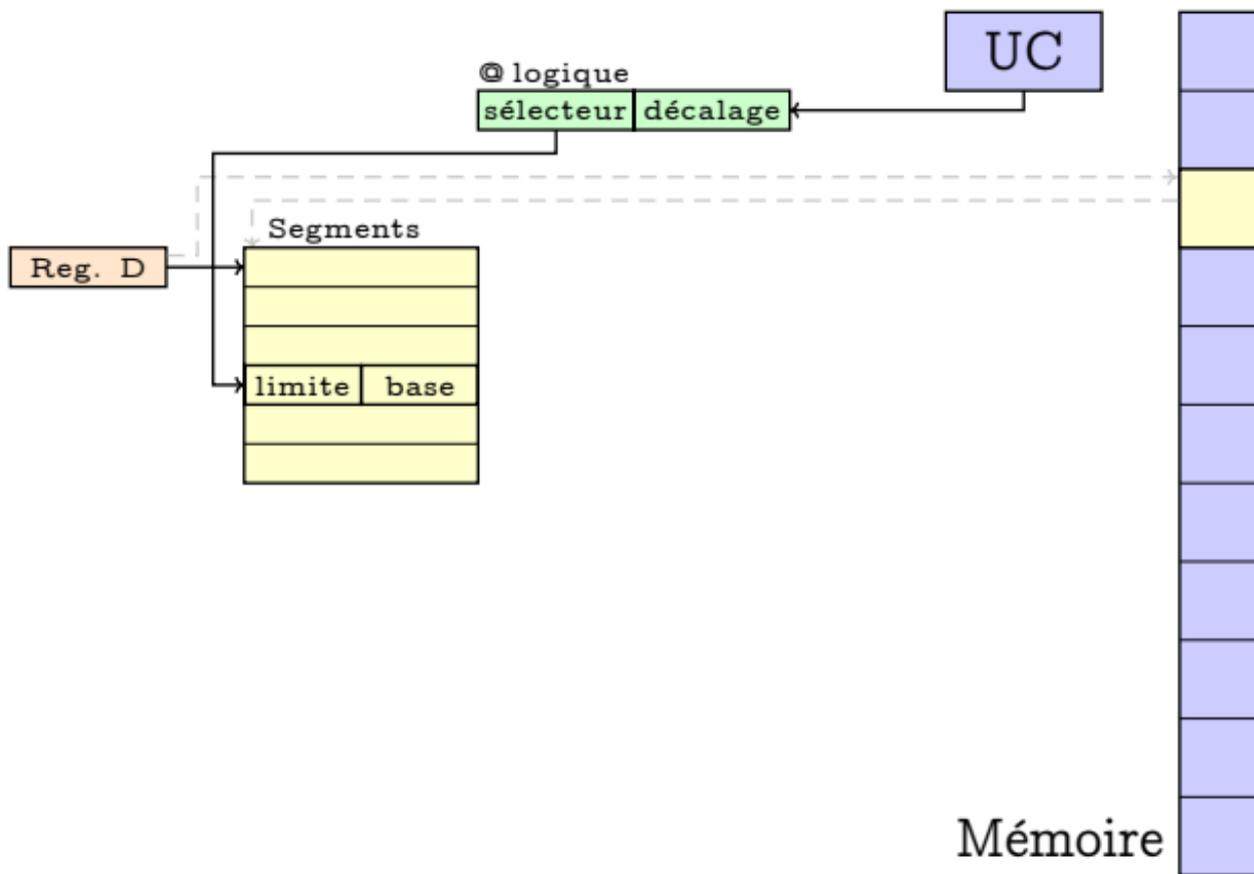


Mémoire

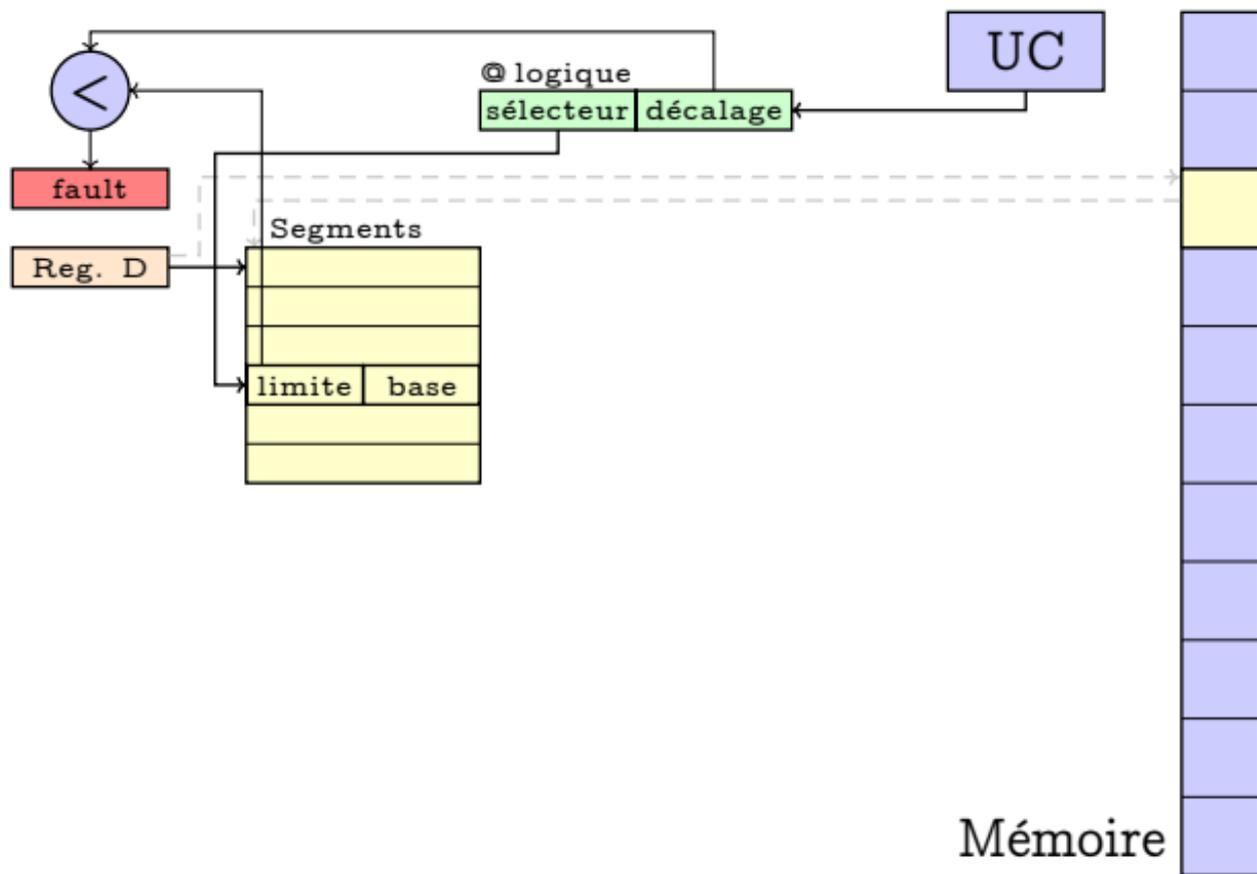
# Segmentation et pagination à deux niveaux



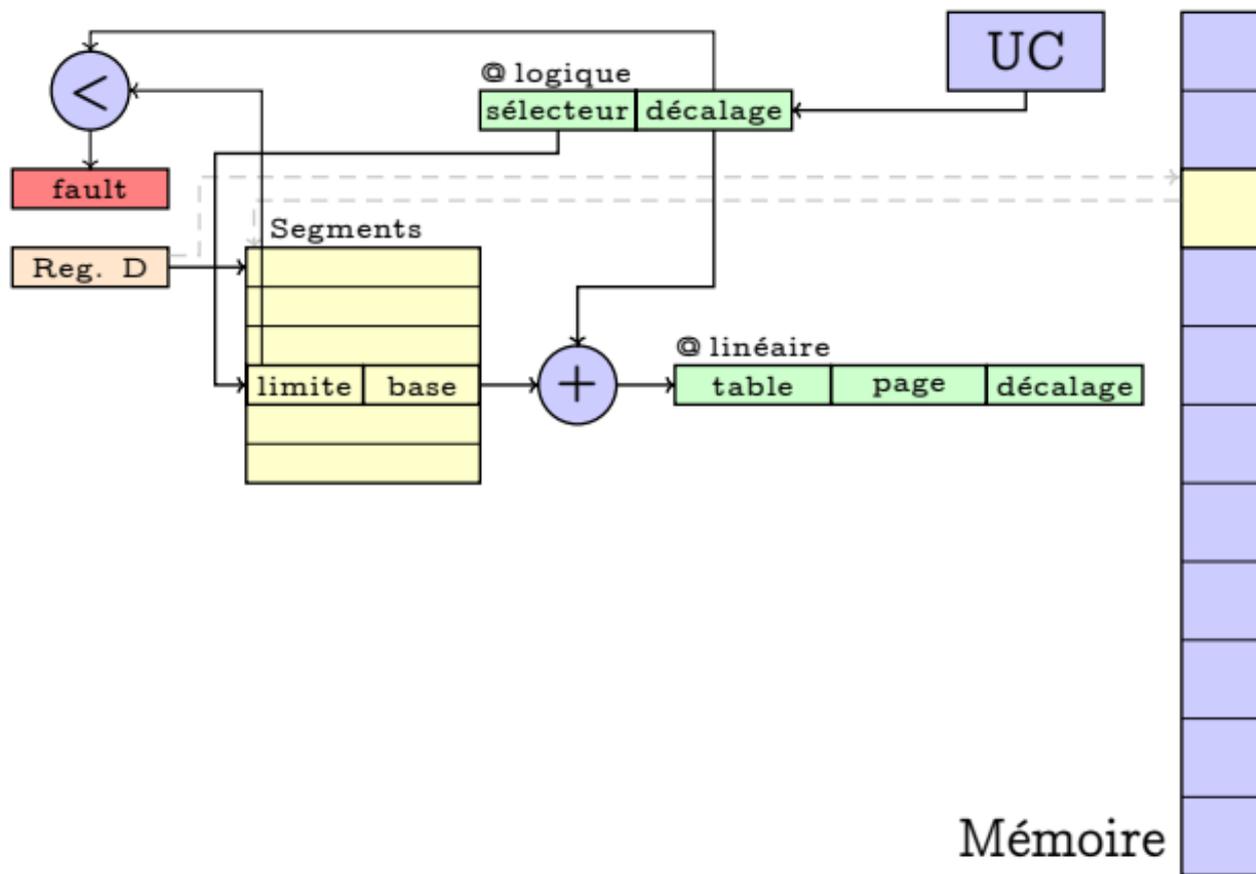
# Segmentation et pagination à deux niveaux



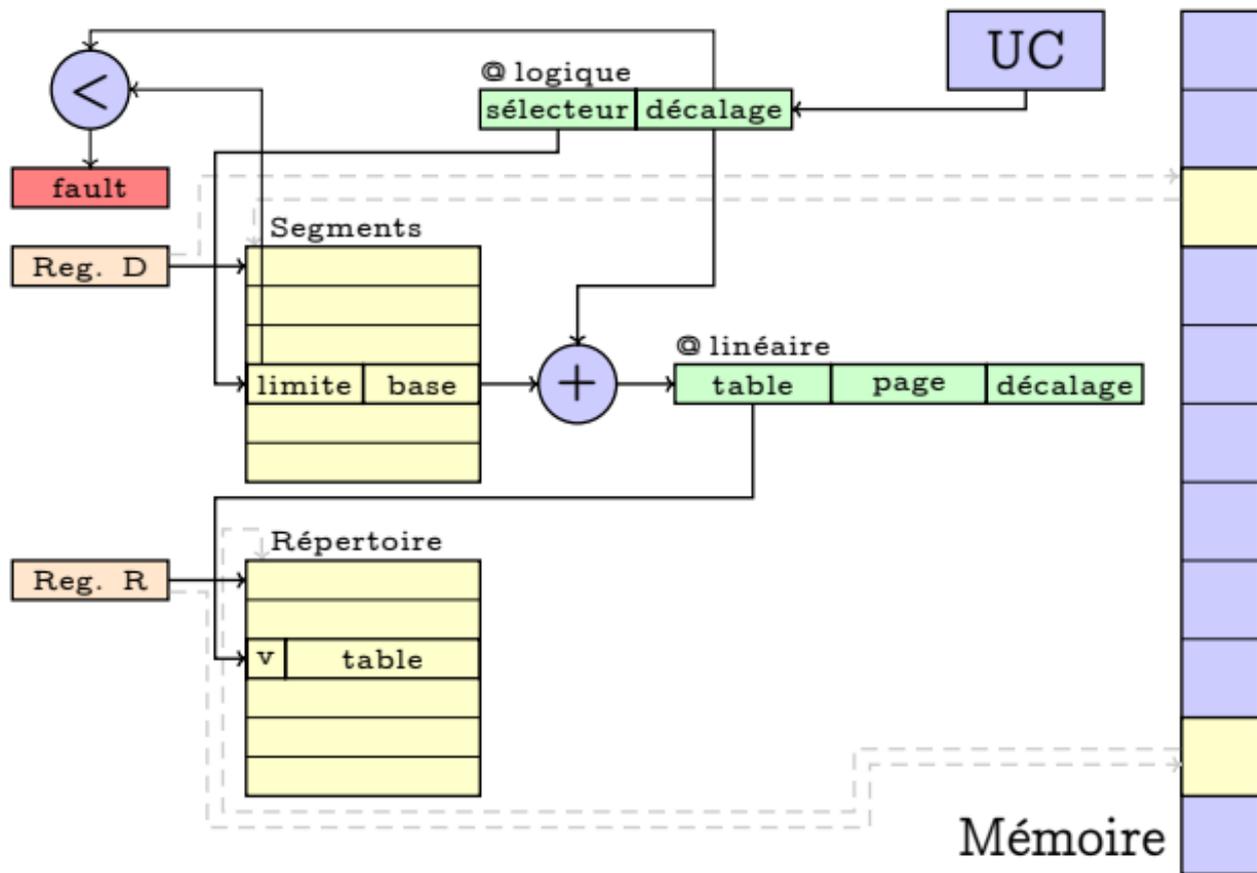
# Segmentation et pagination à deux niveaux



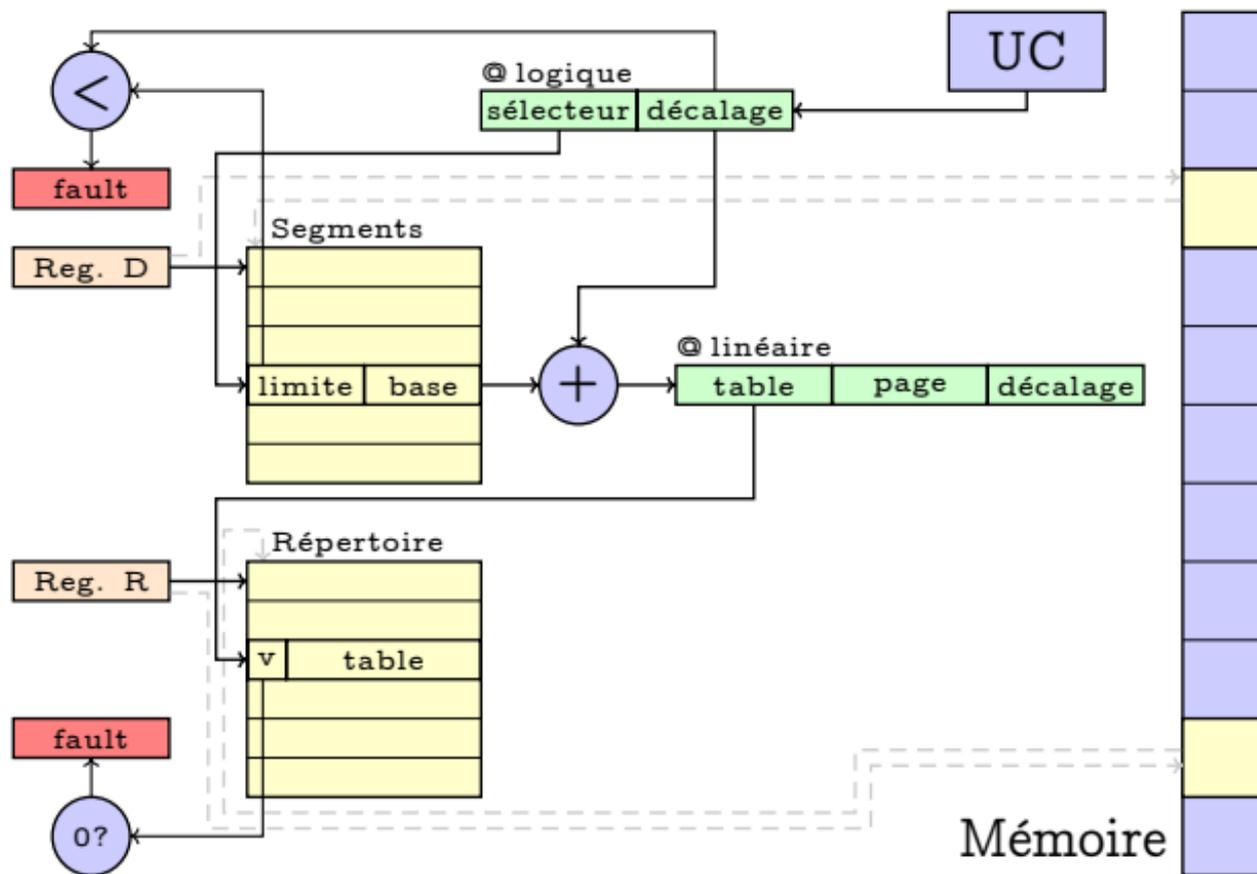
# Segmentation et pagination à deux niveaux



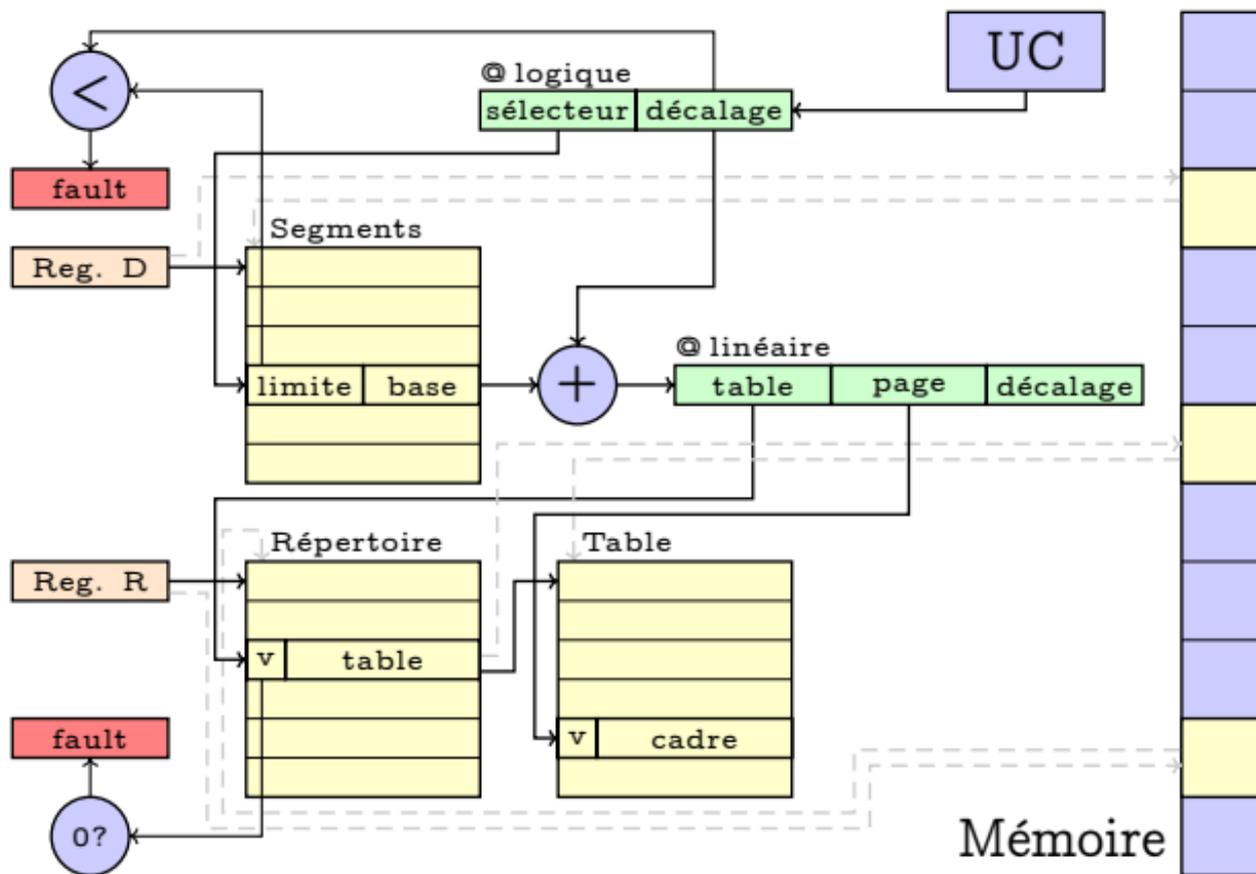
# Segmentation et pagination à deux niveaux



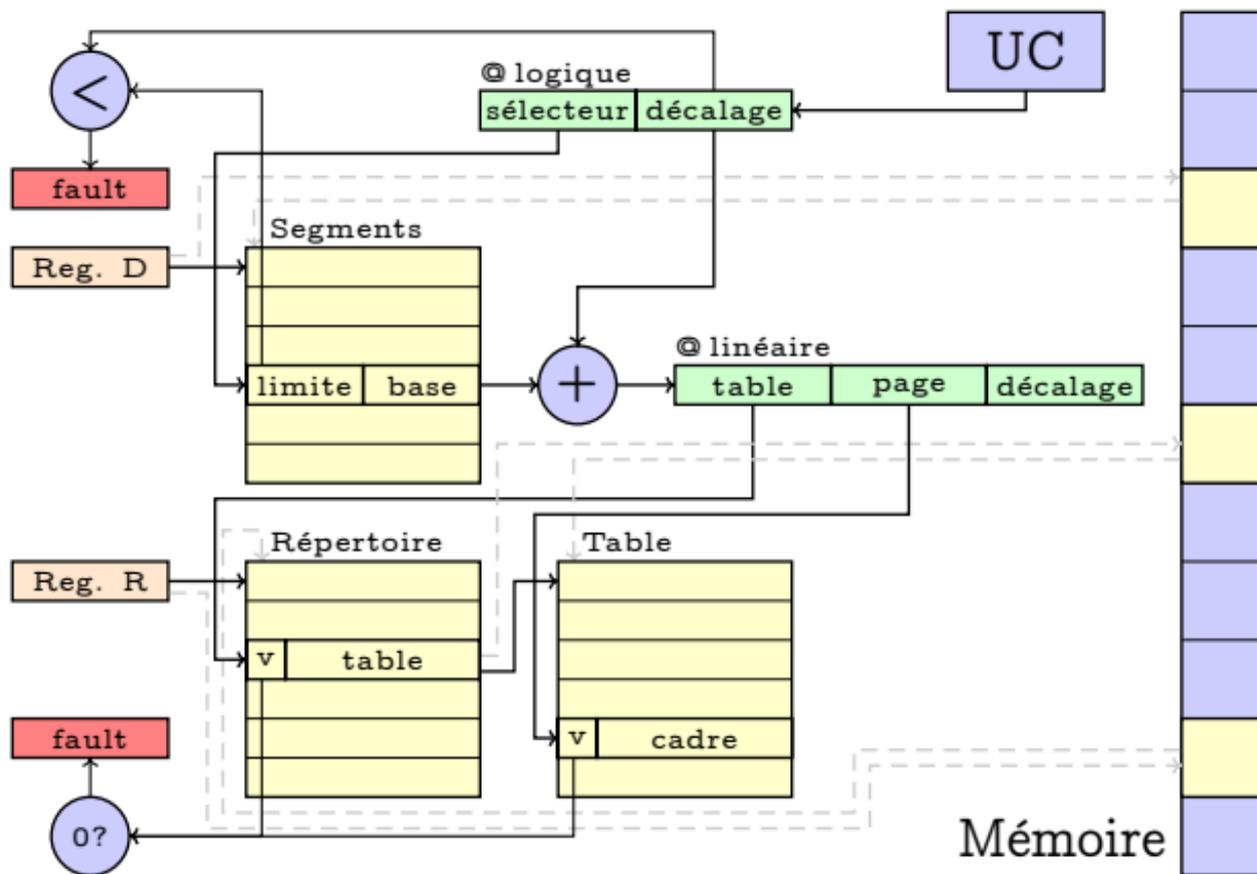
# Segmentation et pagination à deux niveaux



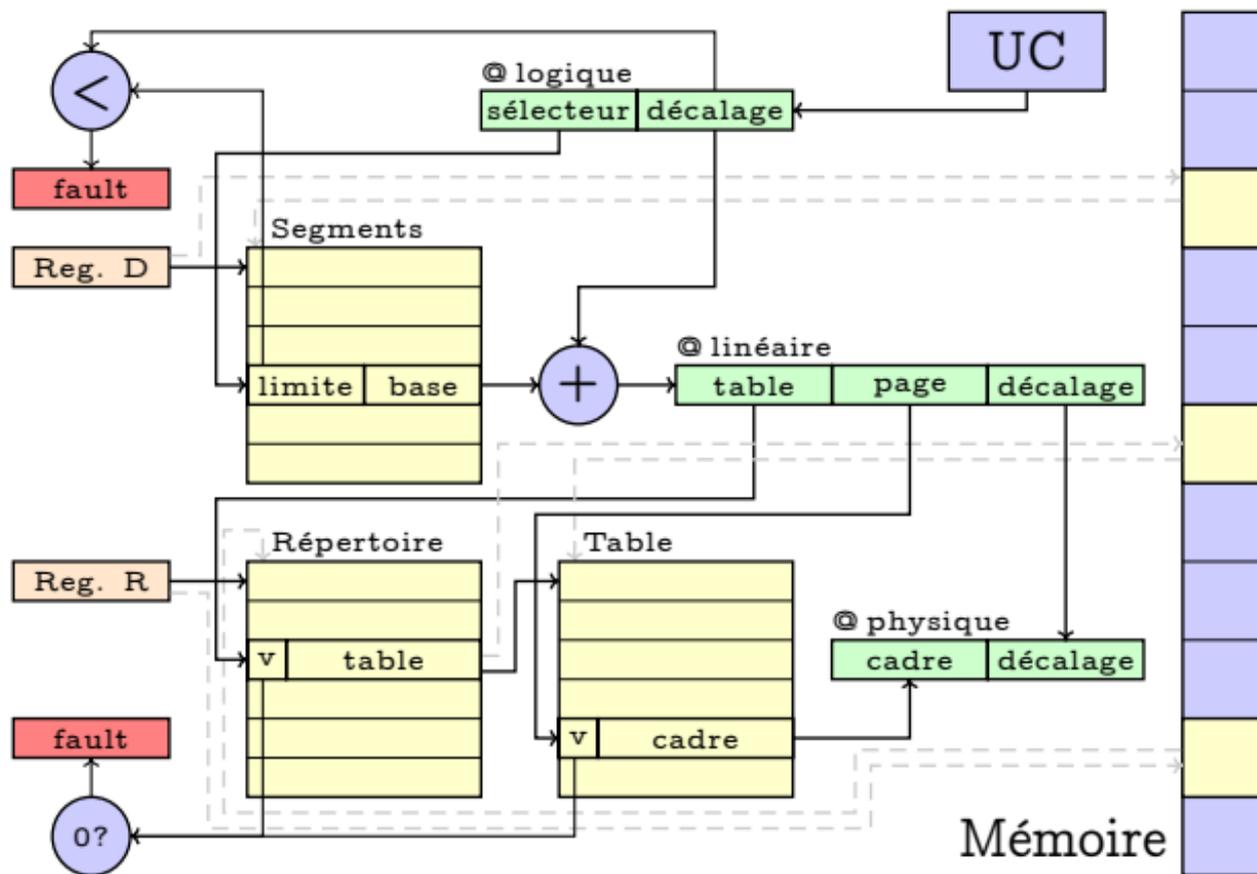
# Segmentation et pagination à deux niveaux



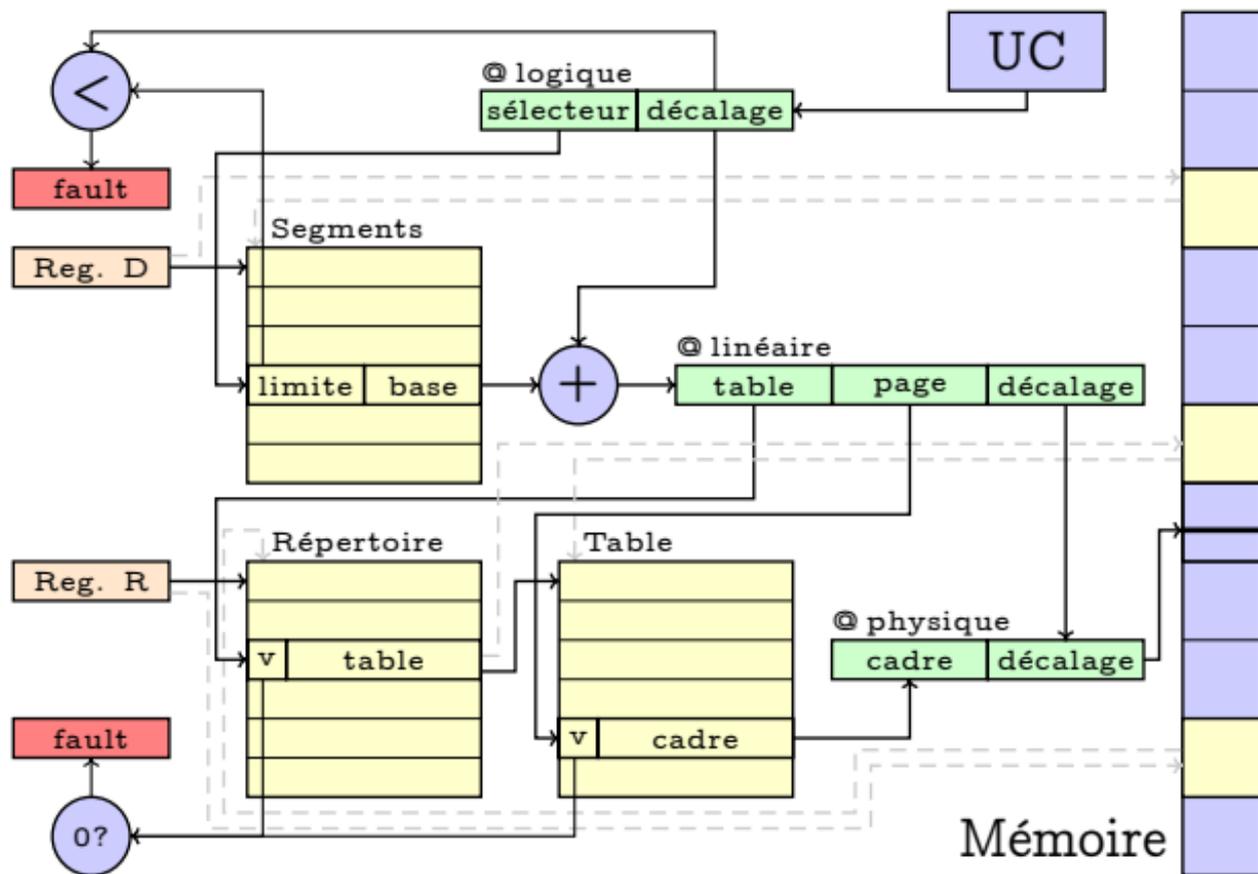
# Segmentation et pagination à deux niveaux



# Segmentation et pagination à deux niveaux



# Segmentation et pagination à deux niveaux



Example

## 32bits

- $2^{16}$  segments de 4Go maximum
- Cadres/Pages de 4Ko ( $m = 12$ )
- Pagination à 2 niveaux ( $n_1 = n_2 = 10$ )

## 64bits

- Plus de segmentation (presque...)
- Adresses virtuelles sur 48bits
- Cadre/Pages de 4Ko ( $m = 12$ )
- Pagination à 4 niveaux ( $n_{1,\dots,4} = 9$ )
- Adresses physiques sur 52bits

Remplacement de pages

## Coût des défauts de pages

- $p$  = probabilité de défaut de page
- $M$  = temps d'accès à la mémoire
- $D$  = temps de traitement du défaut

$$\text{Temps d'accès} = (1-p) \times M + p \times D = M + p(D - M)$$

## Coût des défauts de pages

- $p$  = probabilité de défaut de page
- $M$  = temps d'accès à la mémoire
- $D$  = temps de traitement du défaut

$$\text{Temps d'accès} = (1-p) \times M + p \times D = M + p(D - M)$$

## Propriété

Le temps d'accès à la mémoire est, en moyenne, proportionnel à la probabilité de défaut de page.

## Problème

Combien de cadres (RAM) allouer à chaque processus

## Problème

Combien de cadres (RAM) allouer à chaque processus

## Propriétés

- Plus de cadre → moins de processus  
→ **ralentissement**
- Moins de cadre → plus de défauts  
→ **ralentissement**

## Problème

Combien de cadres (RAM) allouer à chaque processus

## Propriétés

- Plus de cadre → moins de processus  
→ **ralentissement**
- Moins de cadre → plus de défauts  
→ **ralentissement**

## Politiques d'allocation

- Allocation équitable
- Allocation proportionnelle
- Allocation basée sur la priorité

## Données

- $N$  cadres disponibles en RAM
- $P$  processus

## Données

- $N$  cadres disponibles en RAM
- $P$  processus

## Allocation équitable

- Chaque processus reçoit  $N/P$  cadres
- Le reste sert de **tampon** pour supprimer de manière différée

## Données

- $N$  cadres disponibles en RAM
- $P$  processus

## Allocation équitable

- Chaque processus reçoit  $N/P$  cadres
- Le reste sert de **tampon** pour supprimer de manière différée

## Inconvénient

Tous les processus n'ont pas besoin de la même quantité de mémoire...

## Données

- $N$  cadres disponibles en RAM
- $P$  processus
- $\forall i \in [1, P], M_i$  la taille de  $P_i$  (en mémoire virtuelle)

## Données

- $N$  cadres disponibles en RAM
- $P$  processus
- $\forall i \in [1, P]$ ,  $M_i$  la taille de  $P_i$  (en mémoire virtuelle)

## Allocation proportionnelle

- On garde généralement un tampon de  $T$  pages
- $P_i$  reçoit  $(N - T) \times M_i / \sum_i M_i$  cadres

## Données

- $N$  cadres disponibles en RAM
- $P$  processus
- $\forall i \in [1, P]$ ,  $M_i$  la taille de  $P_i$  (en mémoire virtuelle)

## Allocation proportionnelle

- On garde généralement un tampon de  $T$  pages
- $P_i$  reçoit  $(N - T) \times M_i / \sum_i M_i$  cadres

## Inconvénient

Les petits processus font plus de défauts de pages en pratique... *car ils utilisent simultanément une plus grande part de leur code que les gros processus.*

## Données

- $N$  cadres disponibles en RAM
- $P$  processus
- $\forall i \in [1, P]$ ,  $prio_i$  la priorité de  $P_i$

## Données

- $N$  cadres disponibles en RAM
- $P$  processus
- $\forall i \in [1, P]$ ,  $prio_i$  la priorité de  $P_i$

## Allocation basée sur la priorité

- On garde généralement un **tampon** de  $T$  pages
- $P_i$  reçoit  $(N - T) \times prio_i / \sum_i prio_i$  cadres

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple:



Défauts:

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04



Défauts: 0

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04

03	2A	1F	04
----	----	----	----

Défauts: 4

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A

03	2A	1F	04
----	----	----	----

Défauts: 4

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A

03	2A	1F	04
----	----	----	----

Défauts: 4

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A 12

03	2A	1F	04
----	----	----	----

Défauts: 4

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A 12

12	2A	1F	04
----	----	----	----

Défauts: 5

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A 12 03

12	2A	1F	04
----	----	----	----

Défauts: 5

## Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

- Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Exemple: 03 2A 1F 04 2A 12 03

03	2A	1F	04
----	----	----	----

Défauts: 6

## Objectif

Figurer l'état du cache dans le temps pour compter le nombre de défauts

## Objectif

Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Une ligne par cadre de page alloué au processus*

## Objectif

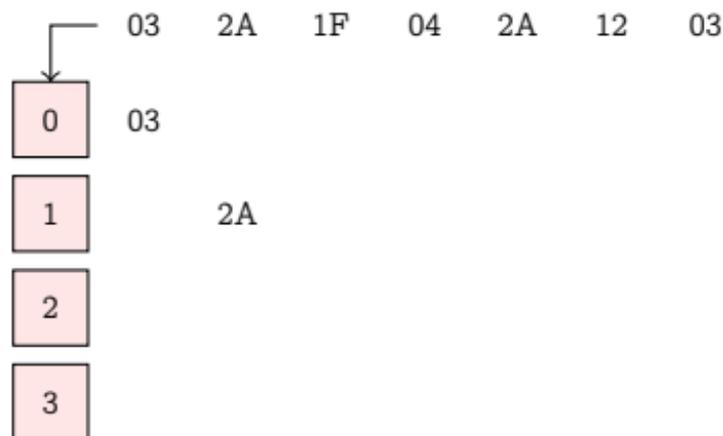
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Une page par colonne*

## Objectif

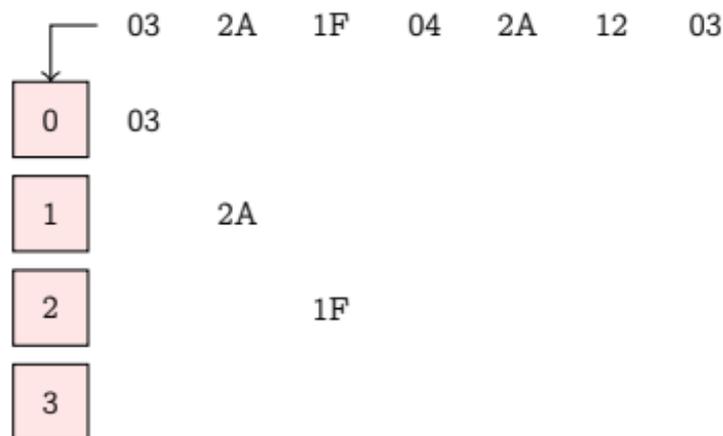
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Une page par colonne*

## Objectif

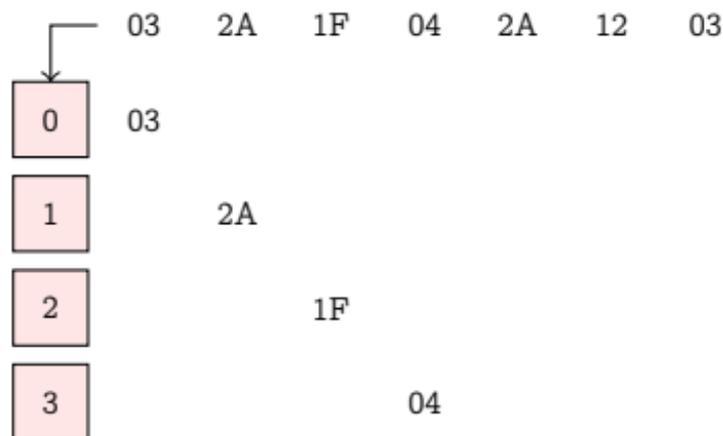
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Une page par colonne*

## Objectif

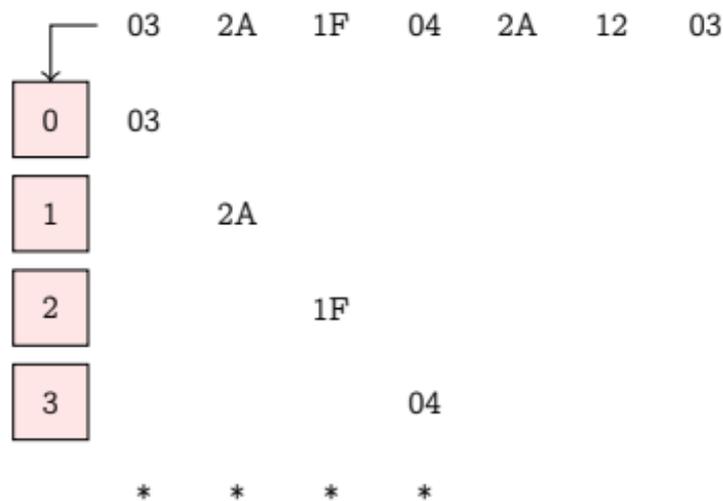
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Une page par colonne*

## Objectif

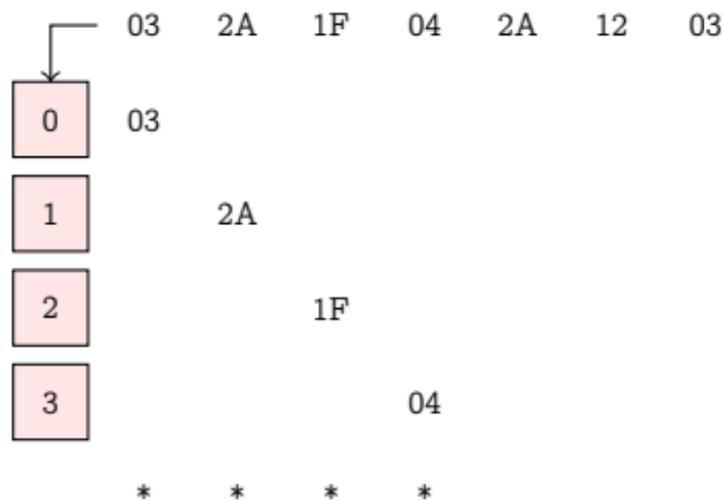
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Marquer les défauts au fur et à mesure*

## Objectif

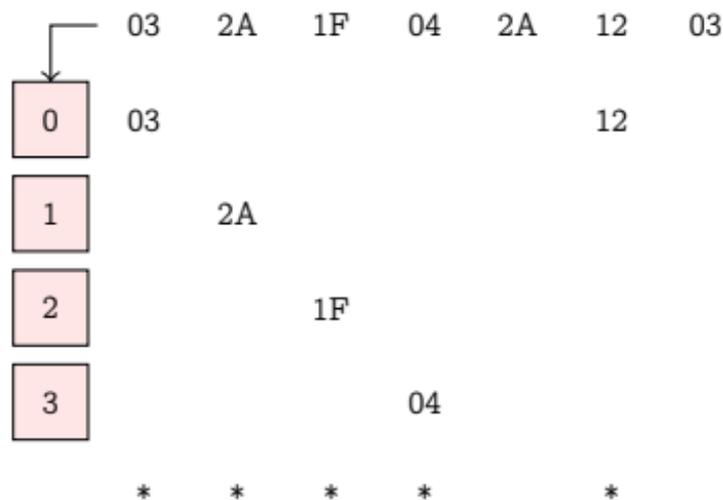
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Pas de défaut* → *laisser en blanc*

## Objectif

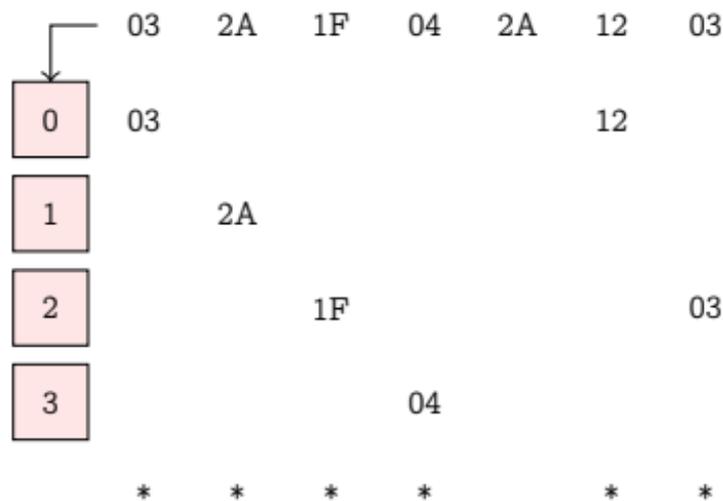
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Et on continue...*

## Objectif

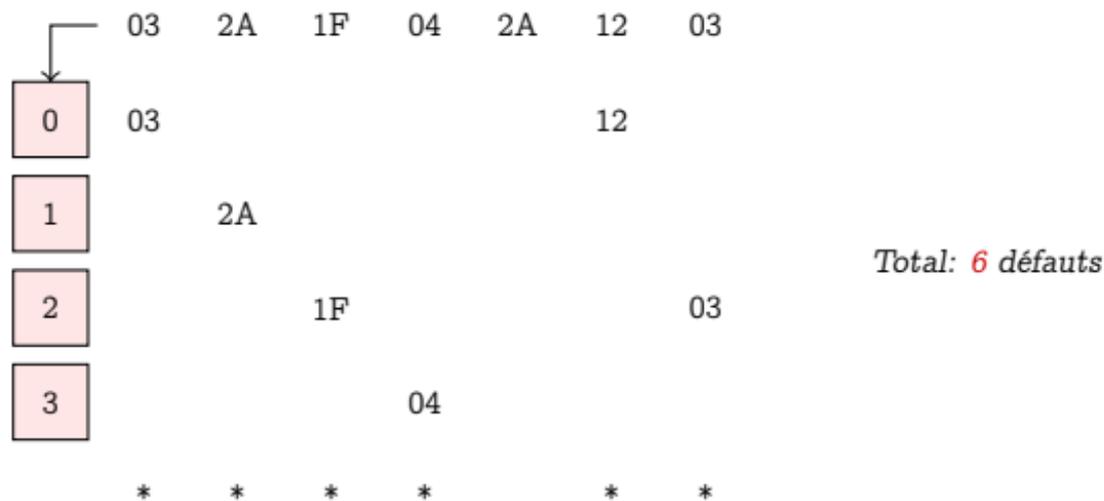
Figurer l'état du cache dans le temps pour compter le nombre de défauts



*Et on continue...*

## Objectif

Figurer l'état du cache dans le temps pour compter le nombre de défauts



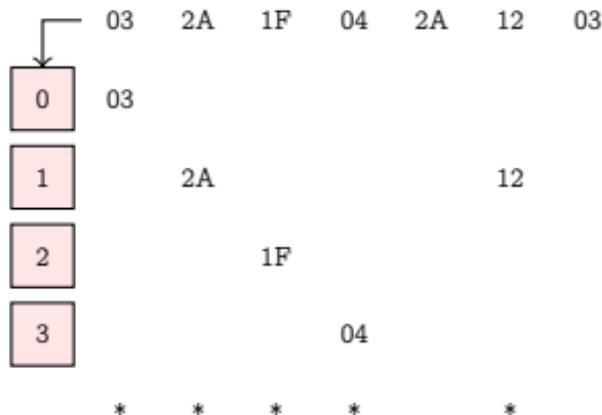
*Et on continue...*

### Politique de remplacement

Pour une instance donnée, il existe une politique de remplacement optimale...

## Politique de remplacement

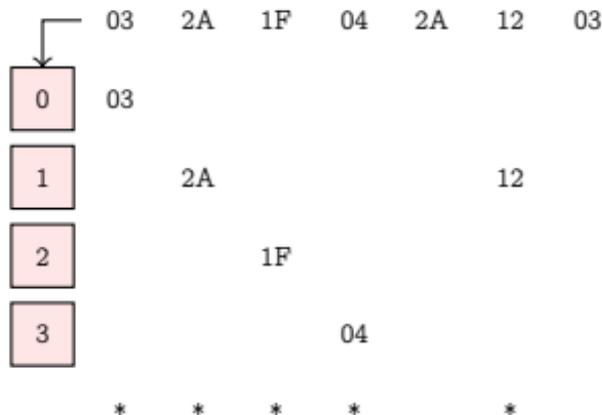
Pour une instance donnée, il existe une politique de remplacement optimale...



Total: 5 défauts

## Politique de remplacement

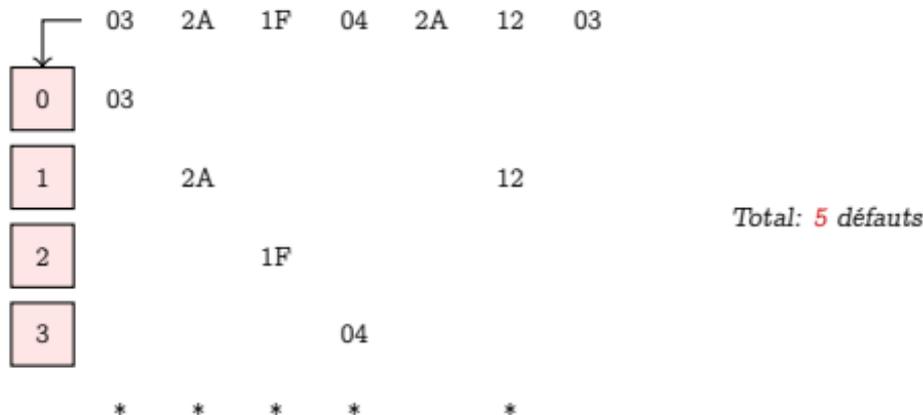
Pour une instance donnée, il existe une politique de remplacement optimale... si on connaît à l'avance les pages qui seront demandées!



Total: 5 défauts

## Politique de remplacement

Pour une instance donnée, il existe une politique de remplacement optimale... si on connaît à l'avance les pages qui seront demandées!



En pratique, on ne connaît pas les pages!

### Algorithme de remplacement

Définir une fonction qui, étant donné l'état actuel (occupation des cadres par des pages), décide quel cadre doit être libéré.

### Algorithme de remplacement

Définir une fonction qui, étant donné l'état actuel (occupation des cadres par des pages), décide quel cadre doit être libéré.

### Classes d'algorithmes

- First In, First Out (FIFO)
  - retirer les pages les plus anciennes
- Basés sur l'utilisation des pages (Least Used)
  - retirer les pages les moins utilisées

## Algorithme de remplacement

Définir une fonction qui, étant donné l'état actuel (occupation des cadres par des pages), décide quel cadre doit être libéré.

## Classes d'algorithmes

- First In, First Out (FIFO)
  - retirer les pages les plus anciennes
- Basés sur l'utilisation des pages (Least Used)
  - retirer les pages les moins utilisées

## Évaluation = nombre de défauts de page

- Sur des instances (en TD, à l'examen)
- Sur des *benchmarks*

# Algorithmes FIFO

File

Retirer la page la plus ancienne

# Principe

File

Retirer la page la plus ancienne

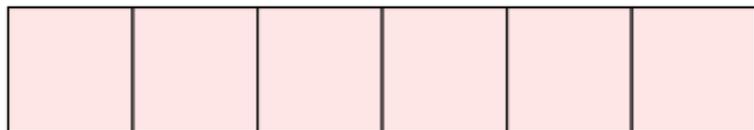
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

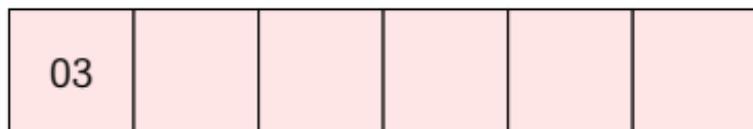


# Principe

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

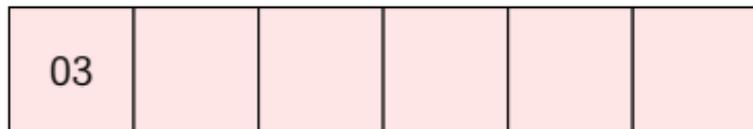


# Principe

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04				
----	----	--	--	--	--

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A			
----	----	----	--	--	--

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	
----	----	----	----	----	--

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	48
----	----	----	----	----	----

# Principe

File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

<del>03</del>	04	2A	1F	12	48
---------------	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	04	2A	1F	12	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	<del>04</del>	2A	1F	12	48
----	---------------	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	<del>2A</del>	1F	12	48
----	----	---------------	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	03	1F	12	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	03	76	12	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	03	76	2A	48
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	03	76	2A	1F
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

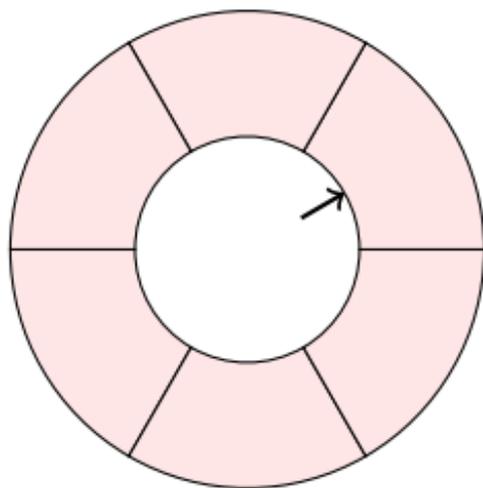
37	B1	03	76	2A	1F
----	----	----	----	----	----

## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

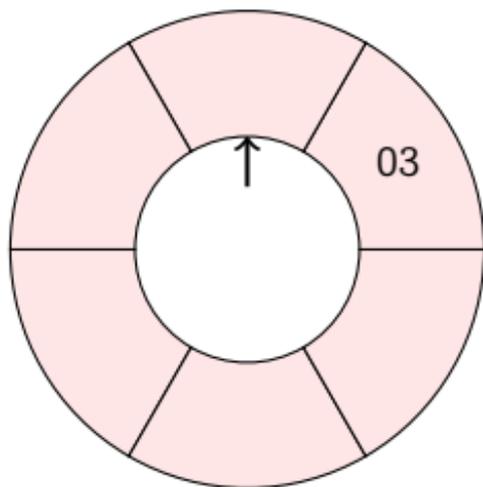


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

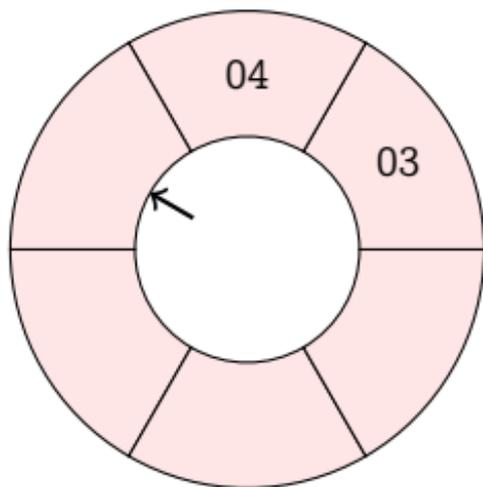


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

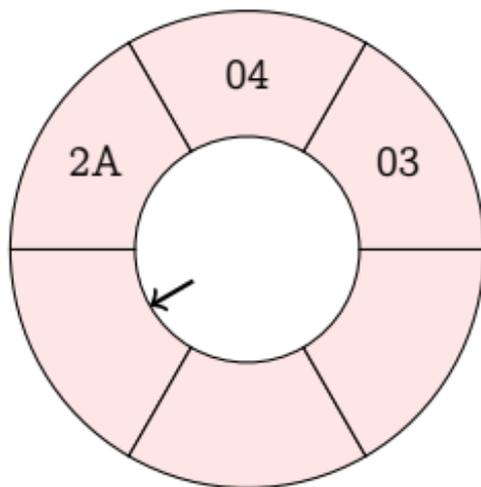


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



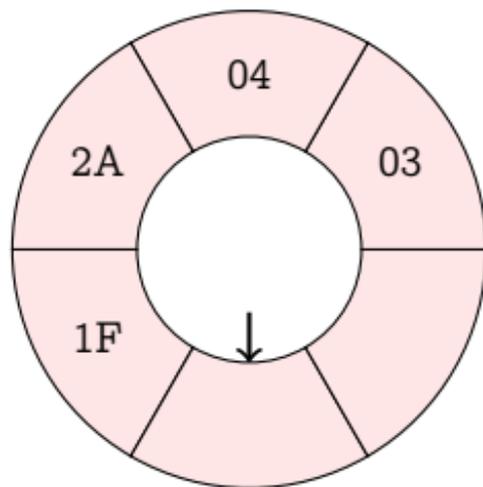
# Principe

## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



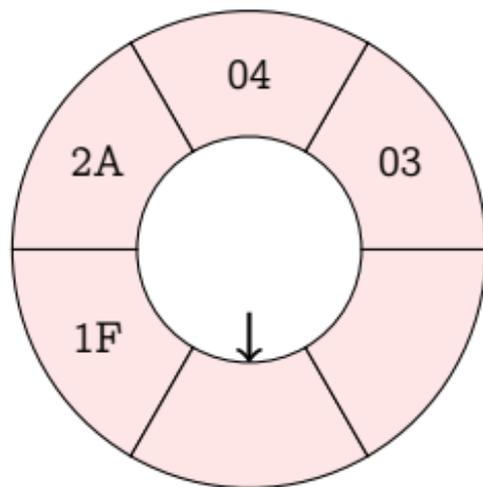
# Principe

## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

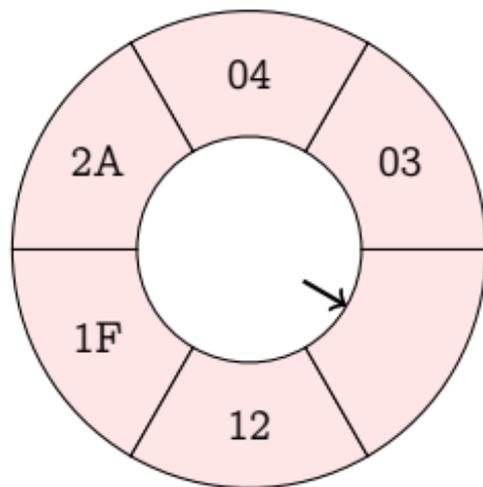


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

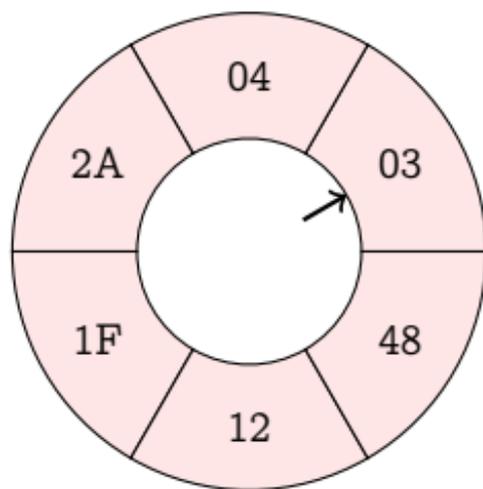


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

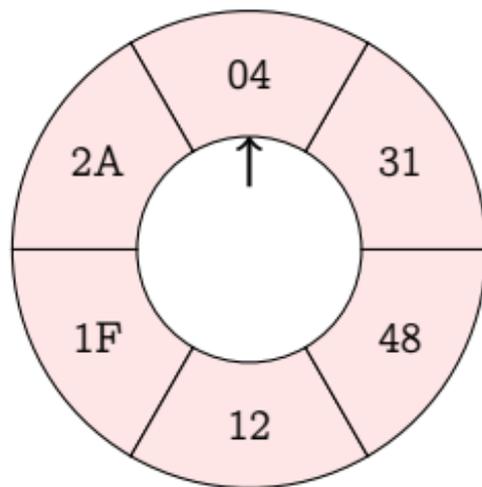


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

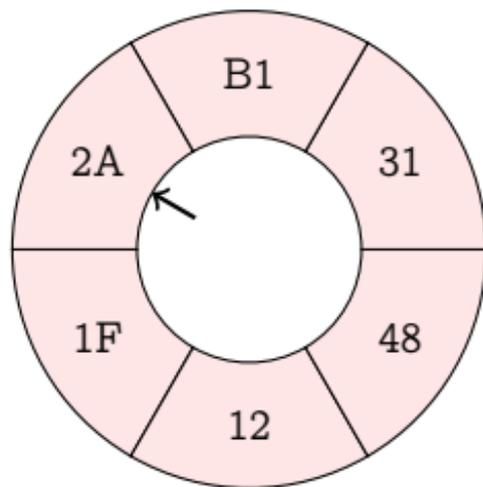


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

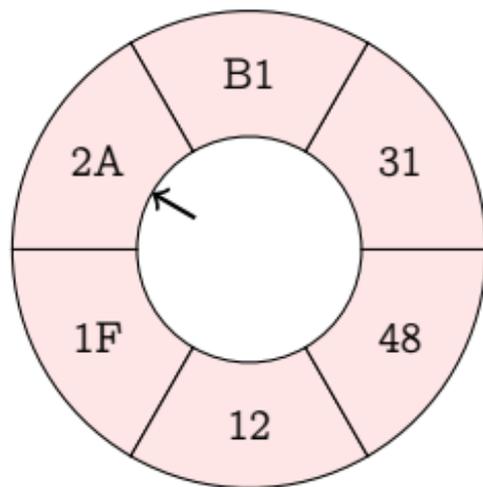


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

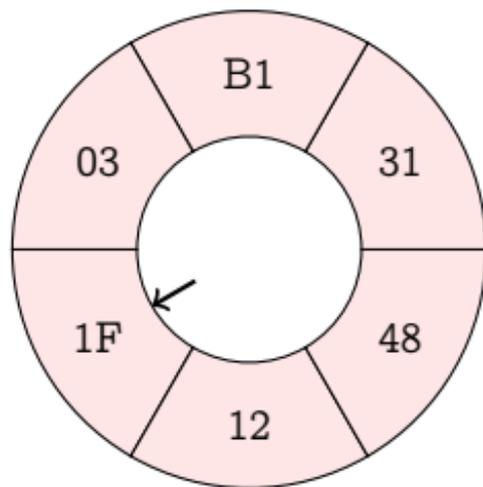


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

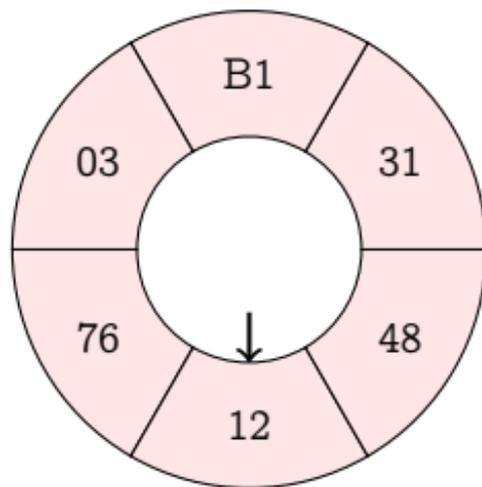


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

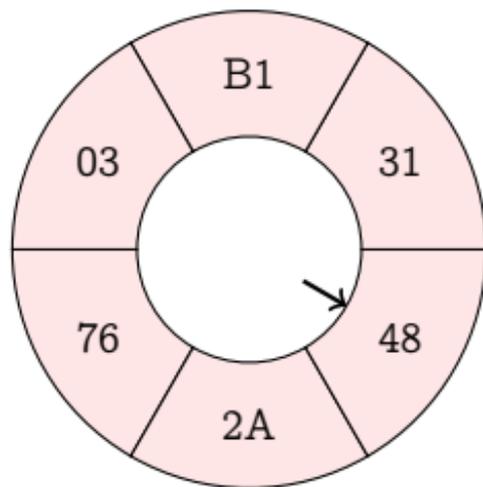


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

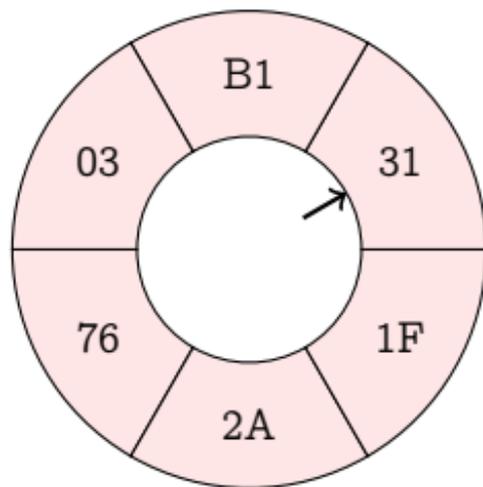


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

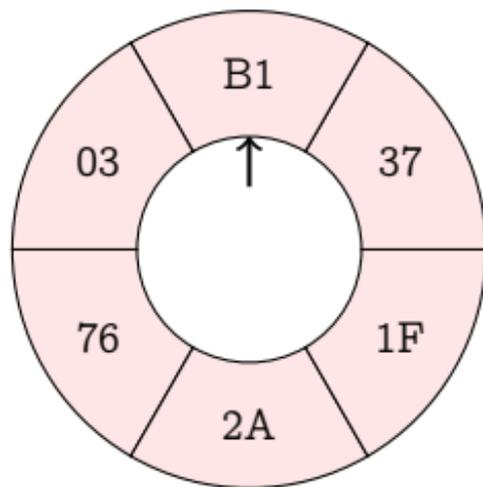


## File

Retirer la page la plus ancienne

- Implémentation = liste circulaire (*clock based*)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

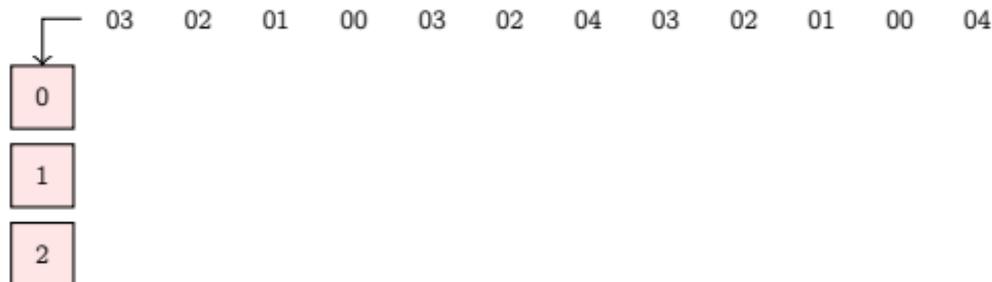


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

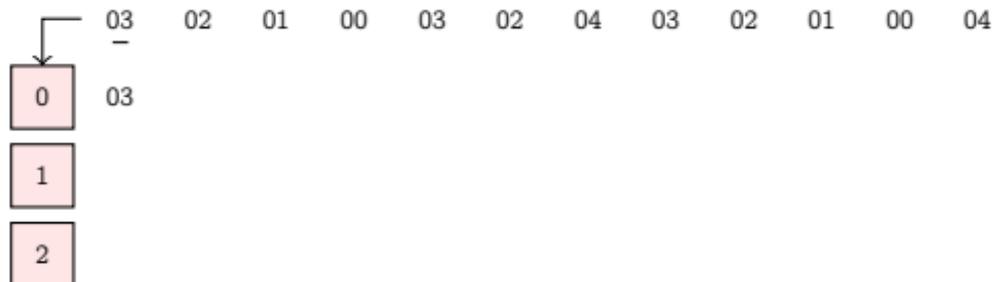


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres



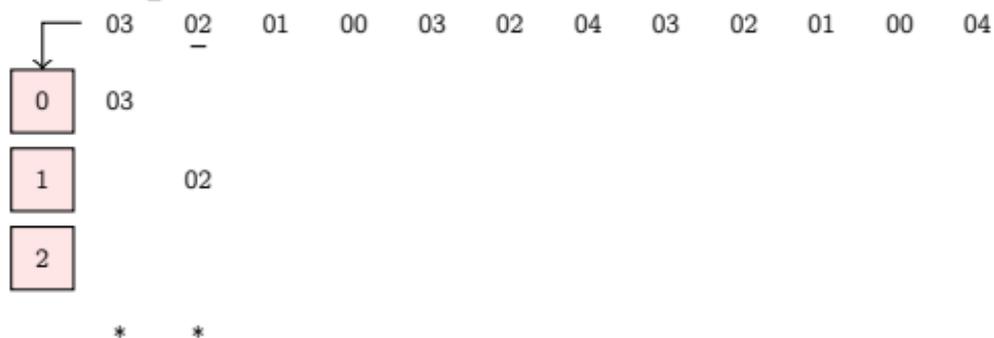
\*

## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

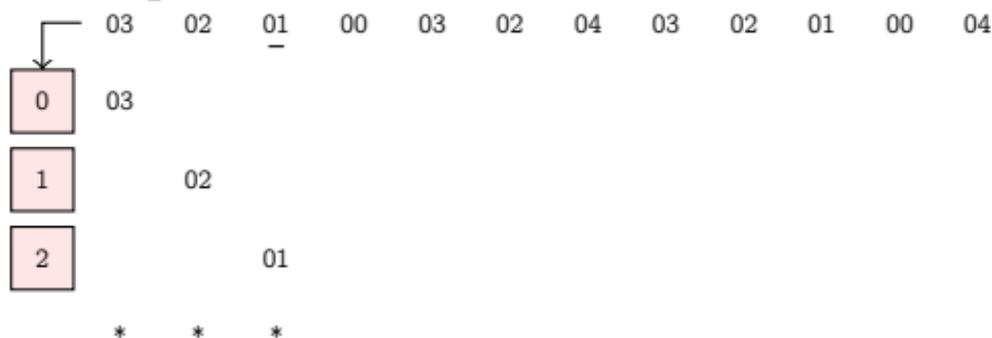


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

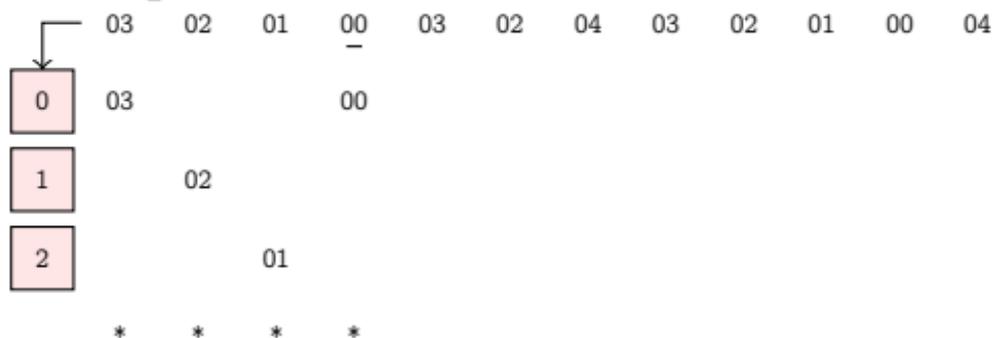


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

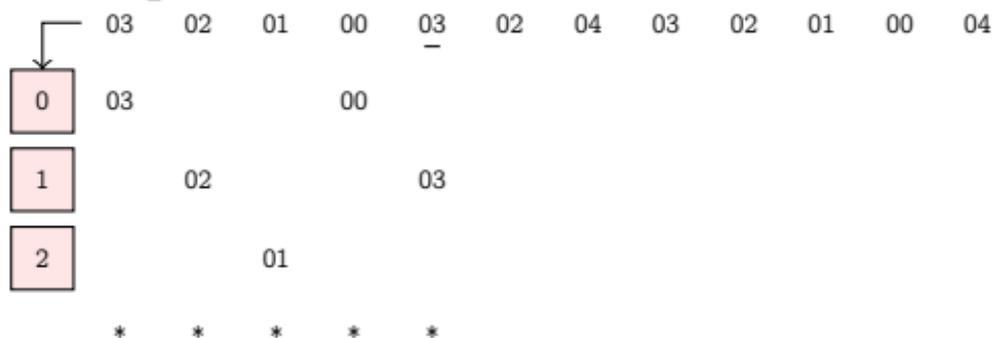


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

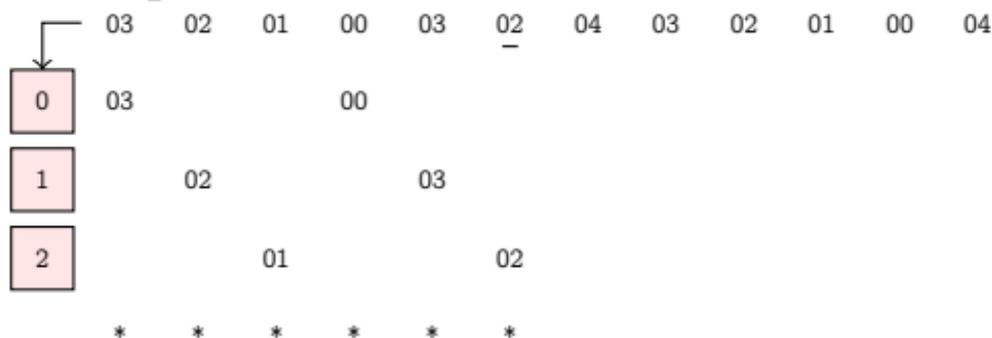


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

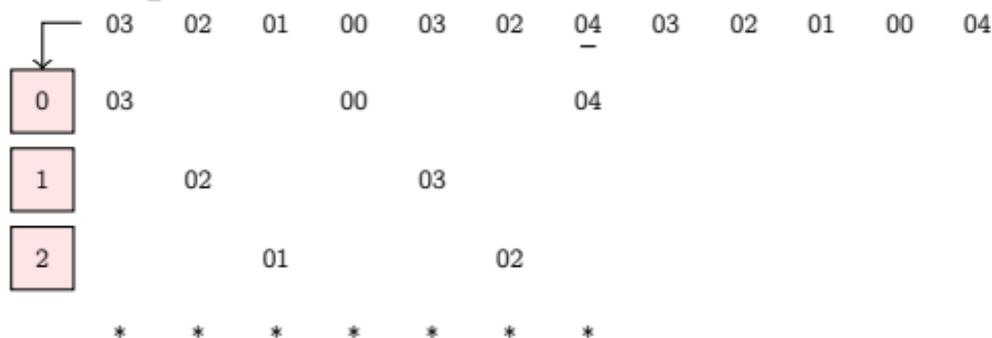


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

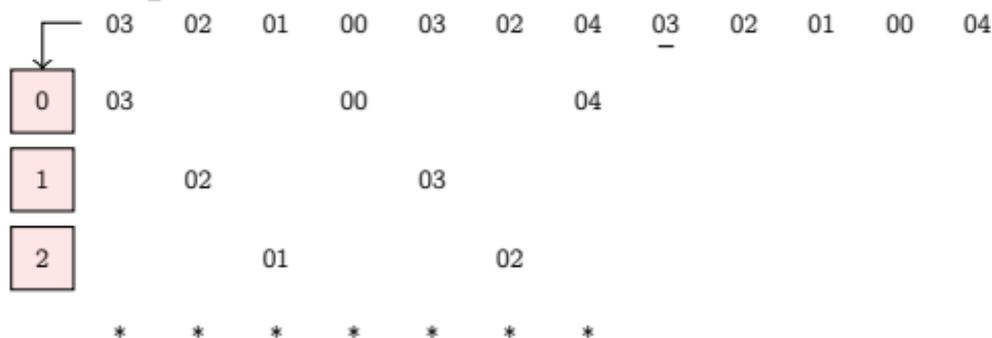


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

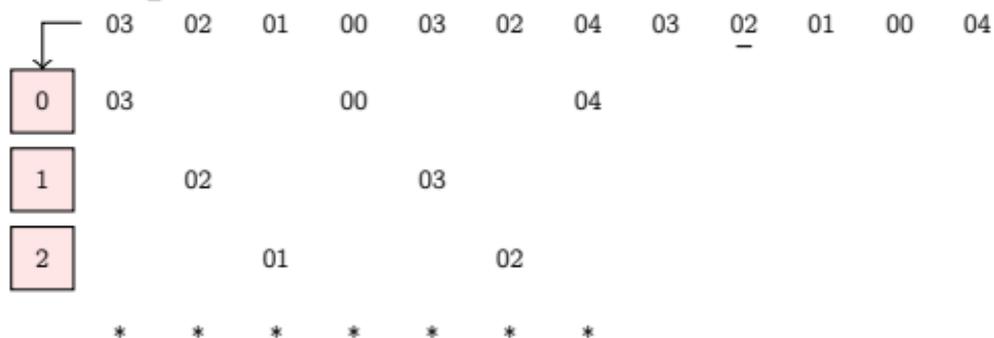


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

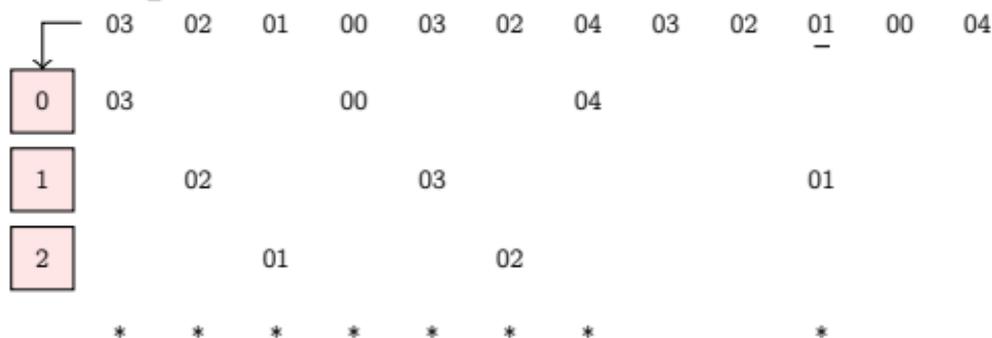


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

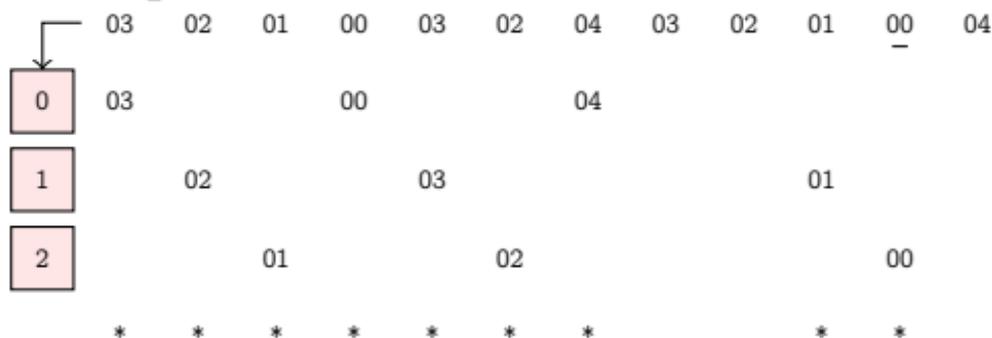


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

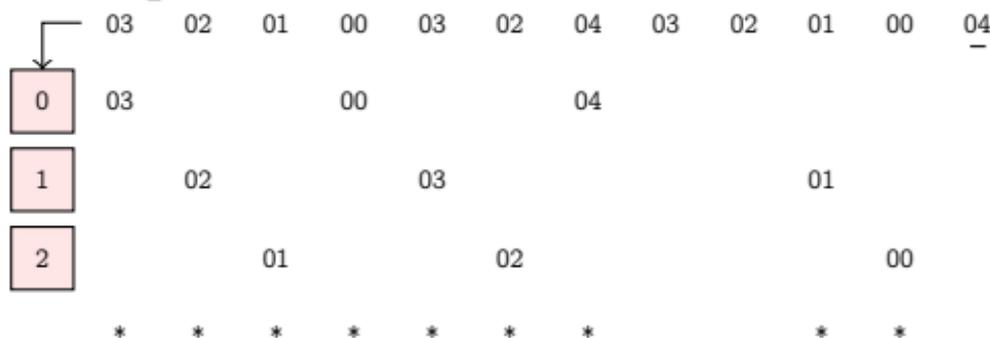


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres

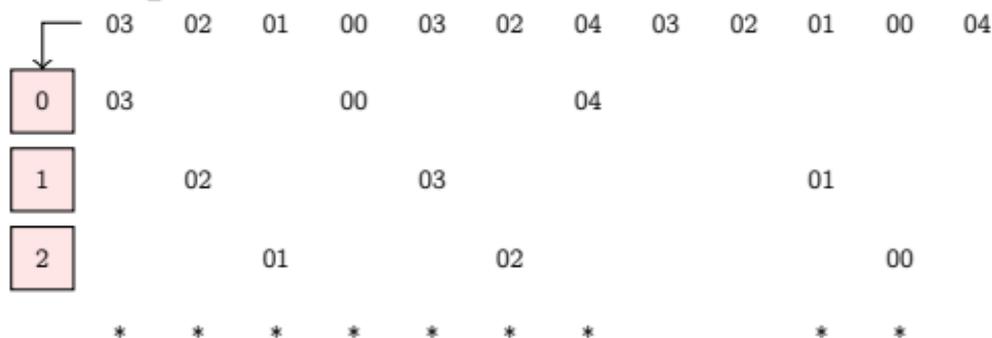


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 3 cadres



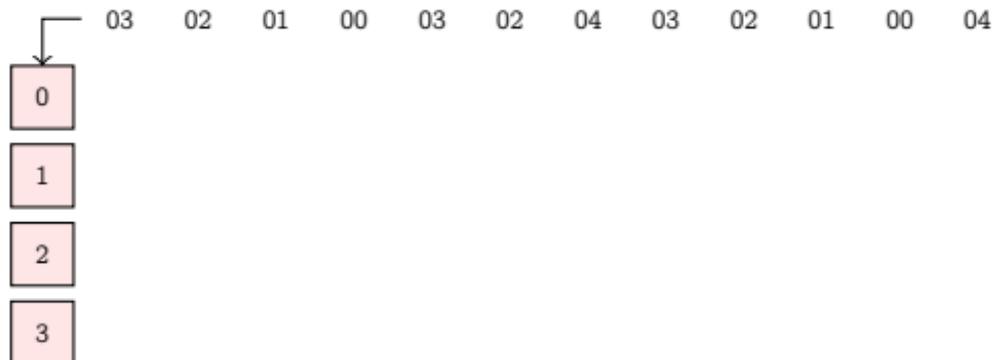
Total: 9 défauts

## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

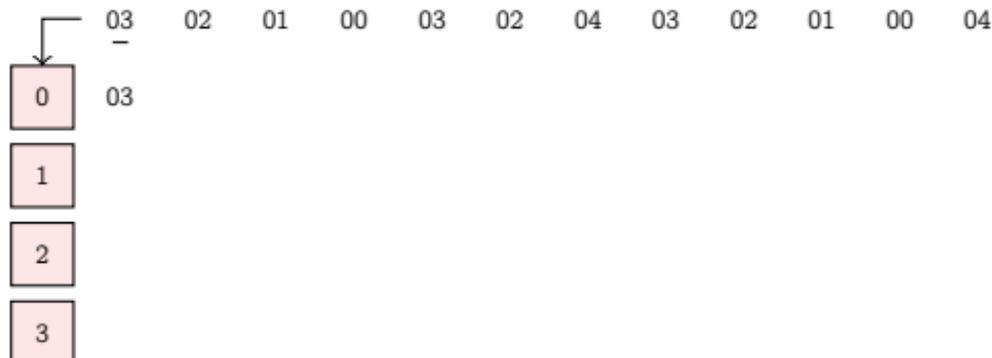


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres



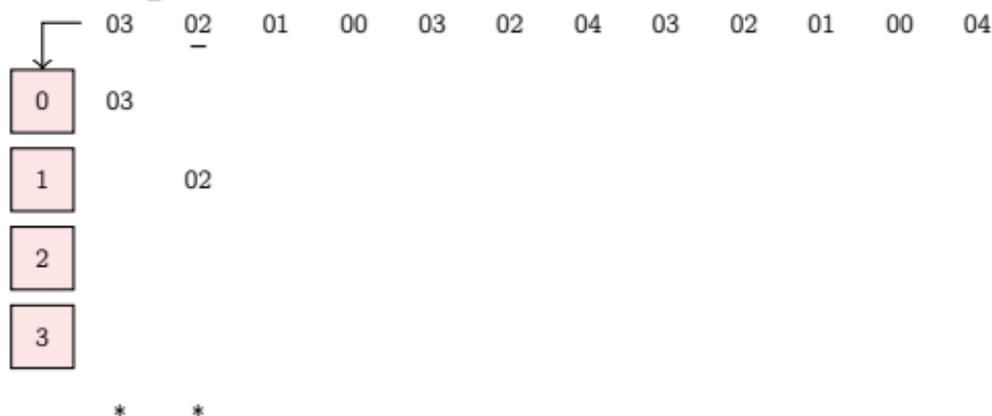
\*

## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres



## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

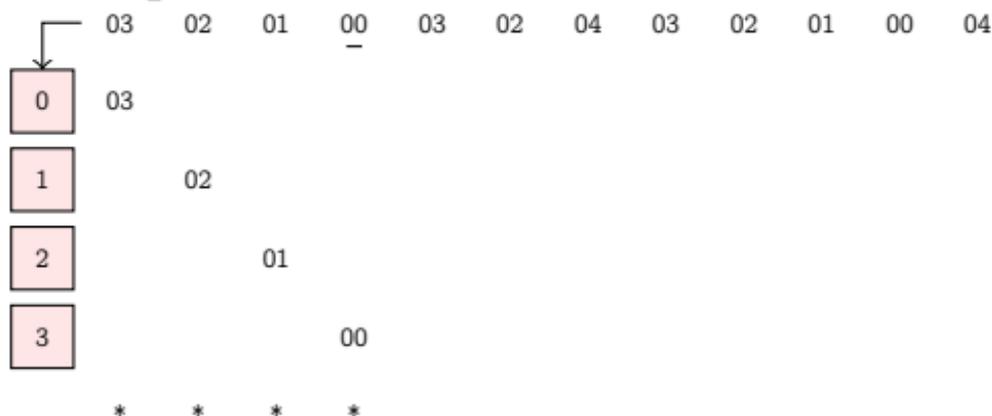


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

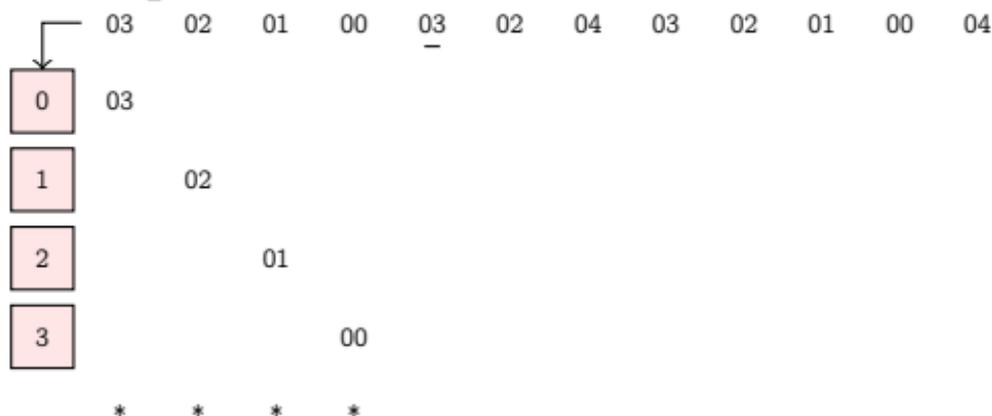


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

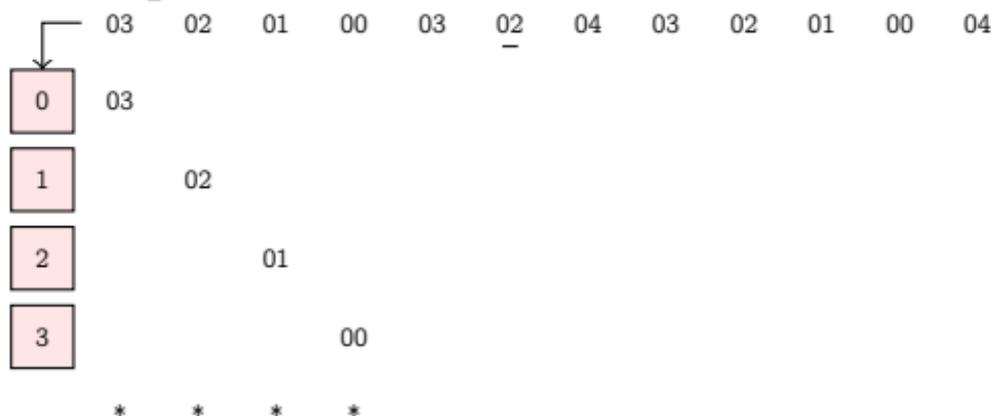


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

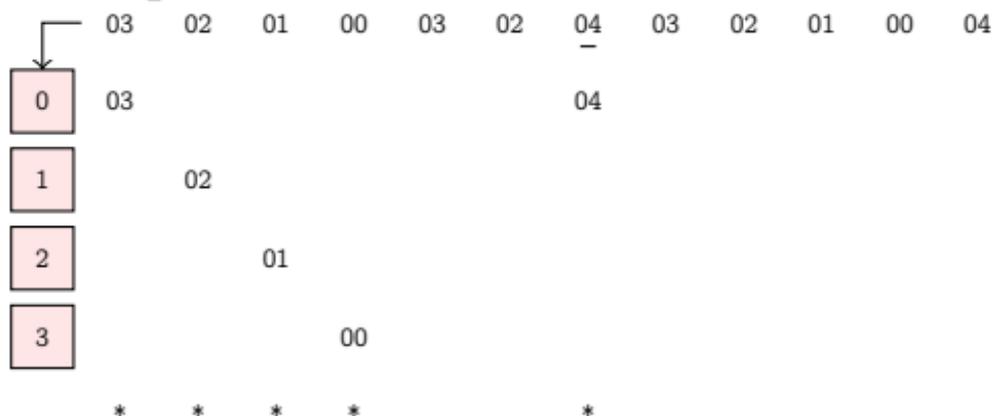


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

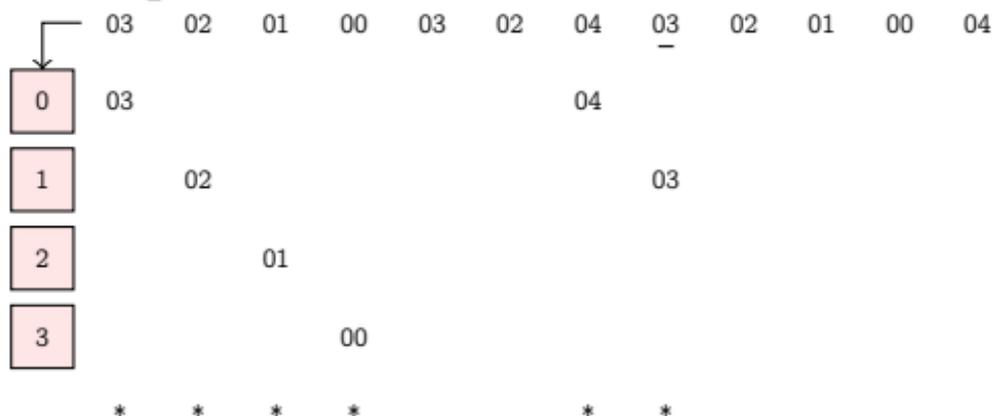


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

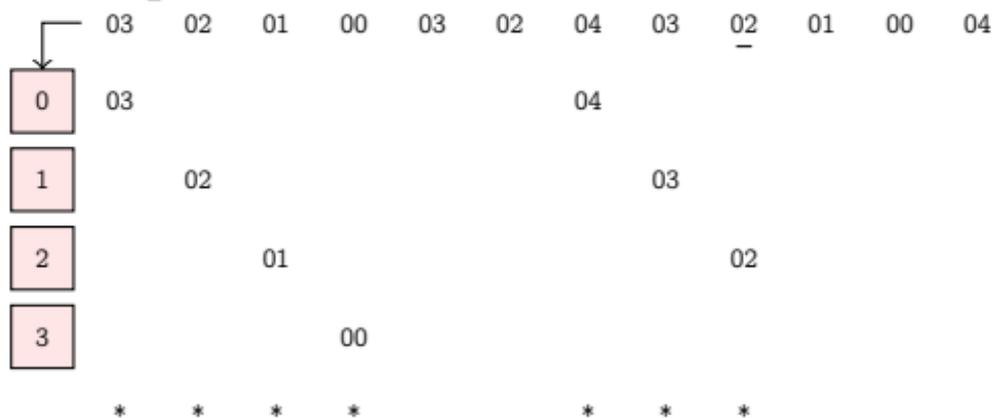


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

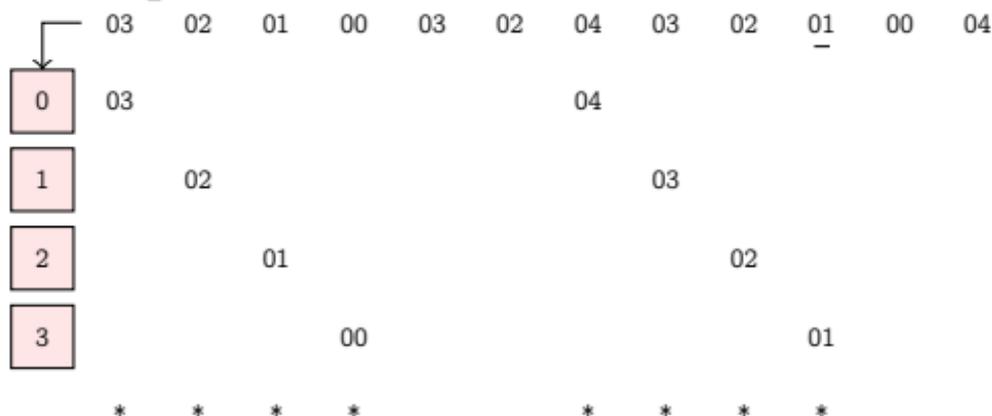


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

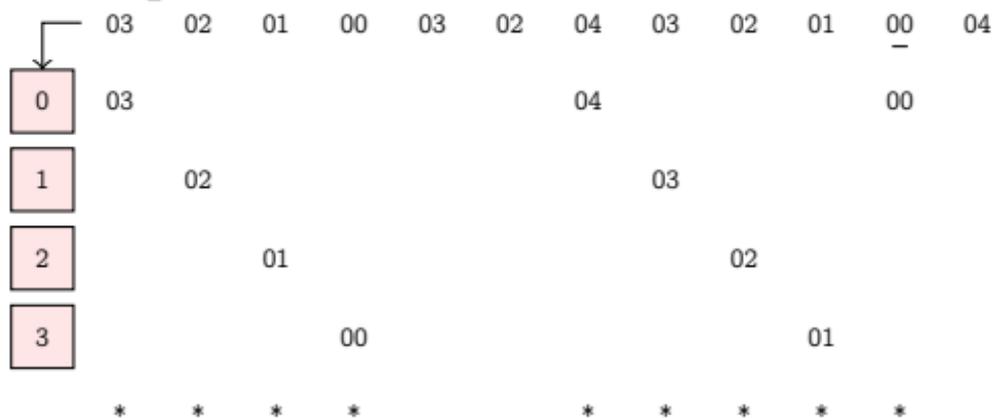


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

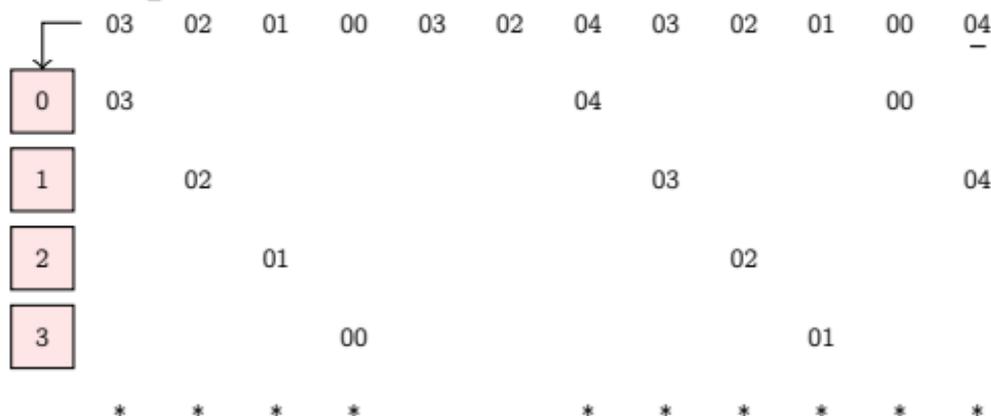


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

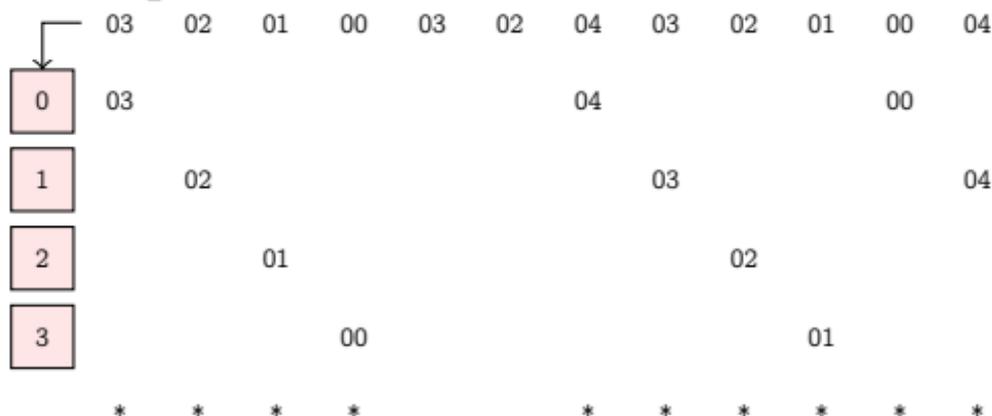


## Algorithmes de type FIFO

*Sur certaines instances...*

Plus de cadres  $\rightarrow$  plus de défauts!

Exemple: avec 4 cadres

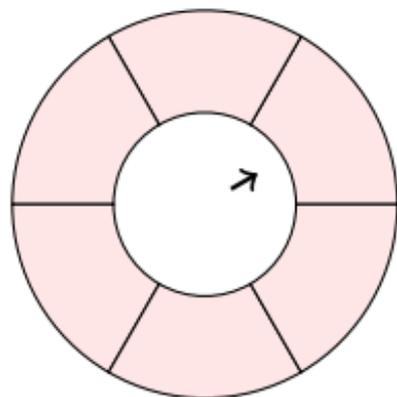


Total: 10 défauts

### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

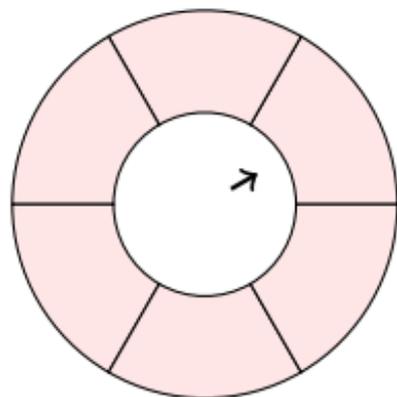
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

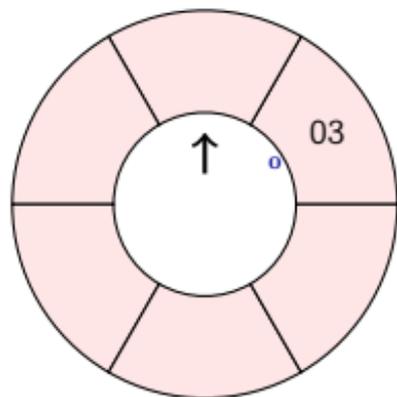
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

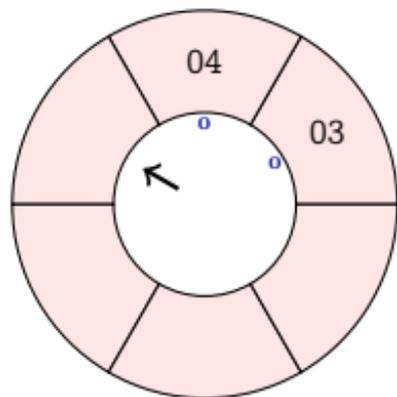
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

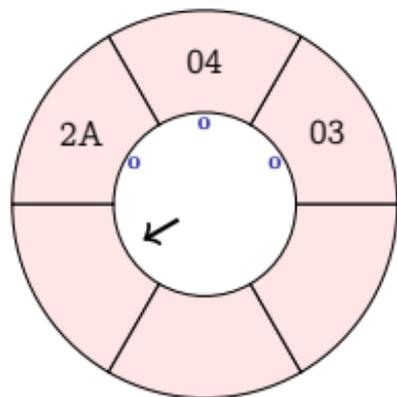
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

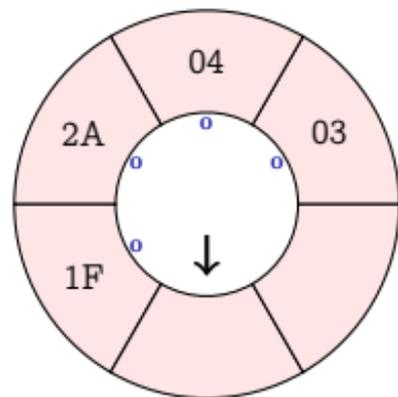
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

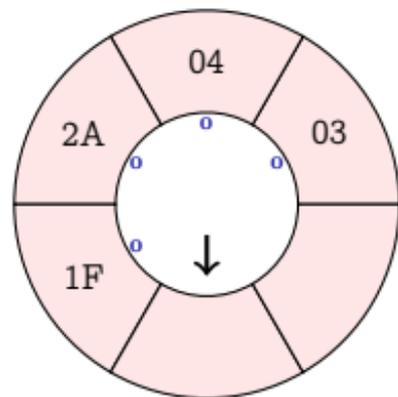
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

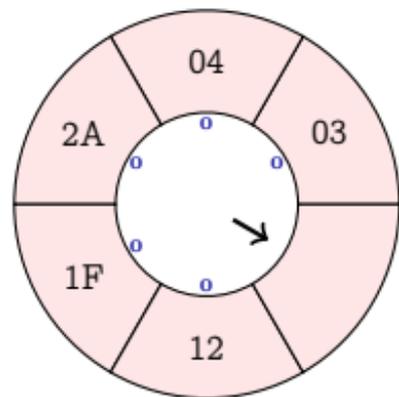
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

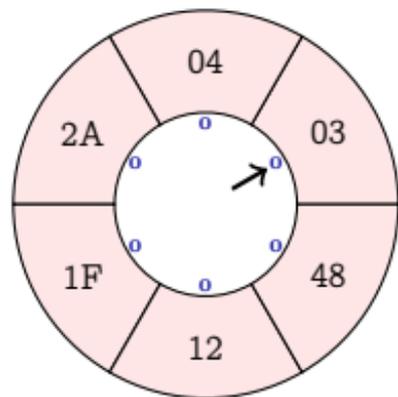
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

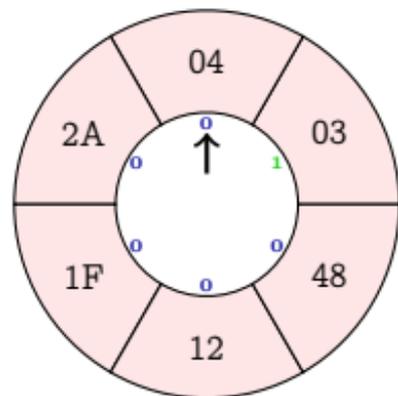
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

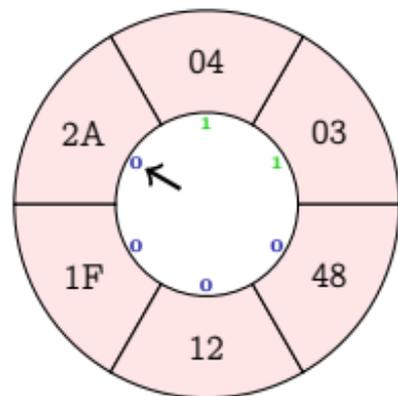
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

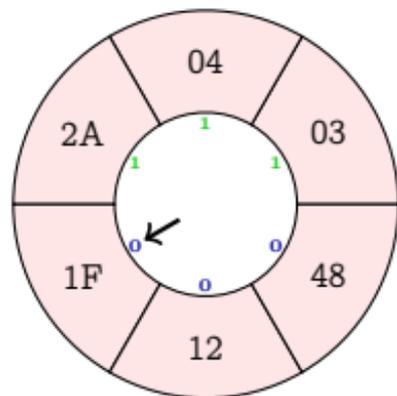
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

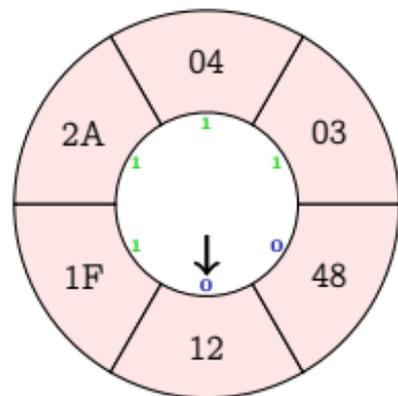
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

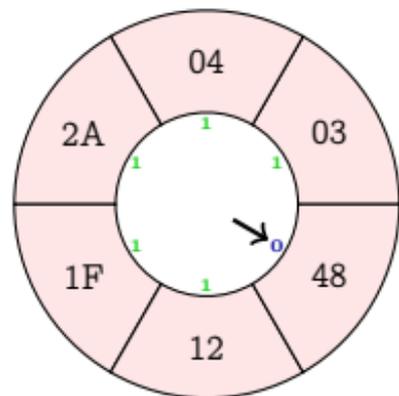
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

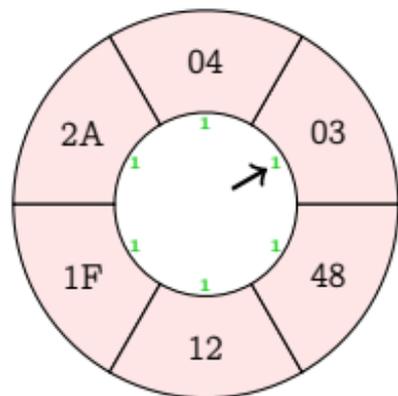
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



### Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

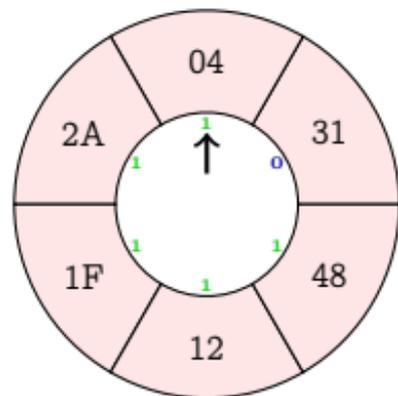
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

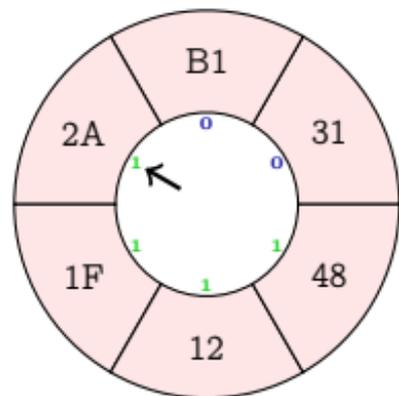
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

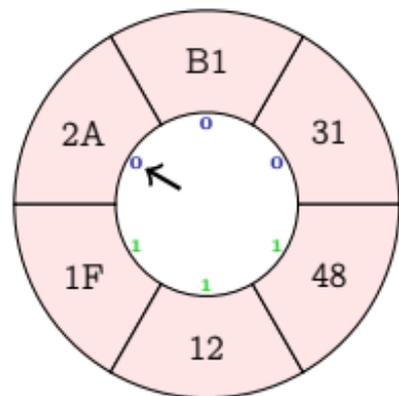
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

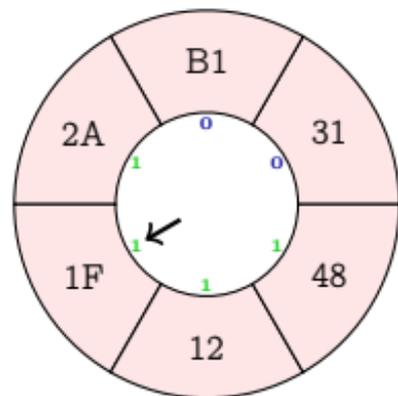
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

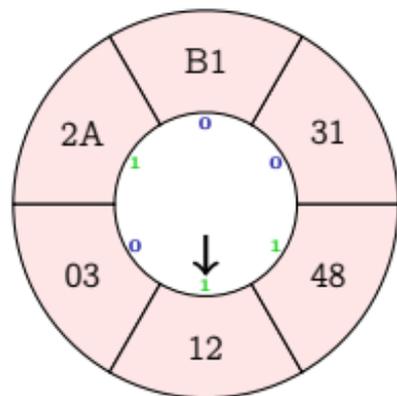
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

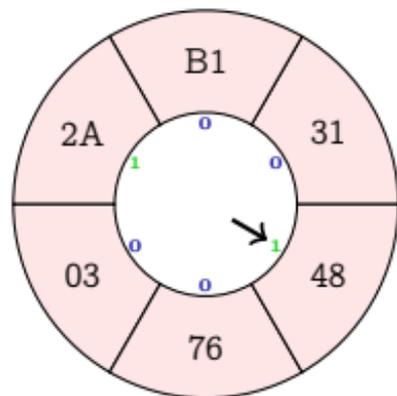
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

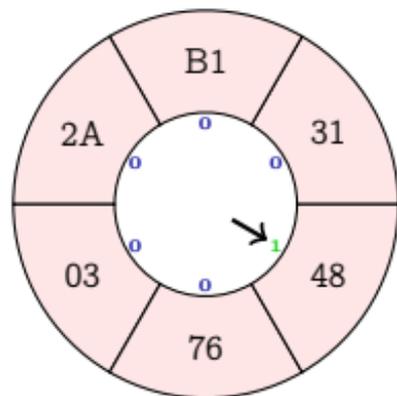
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

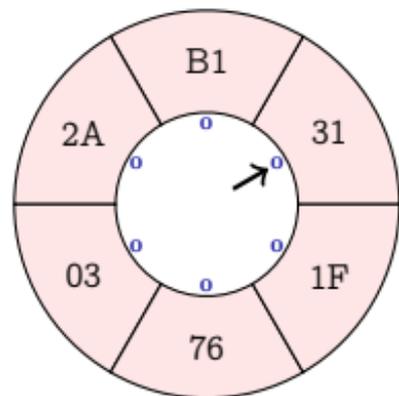
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

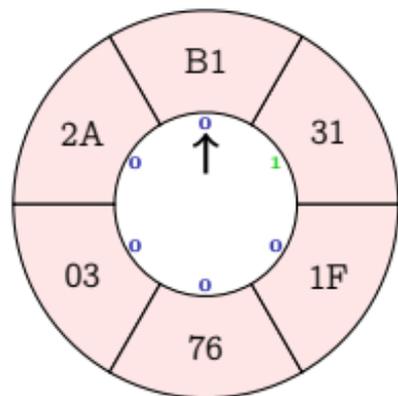
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

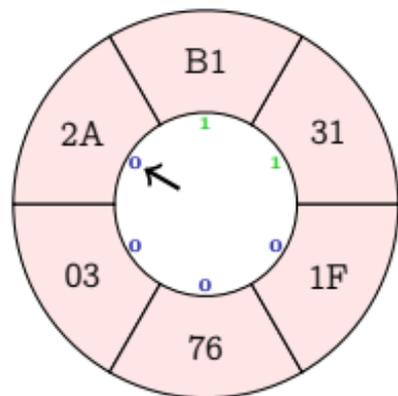
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

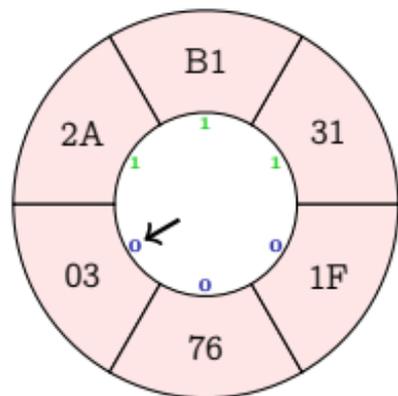
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

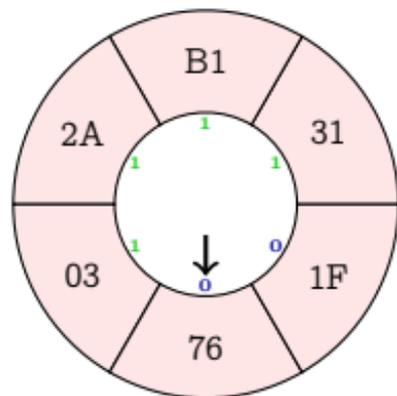
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

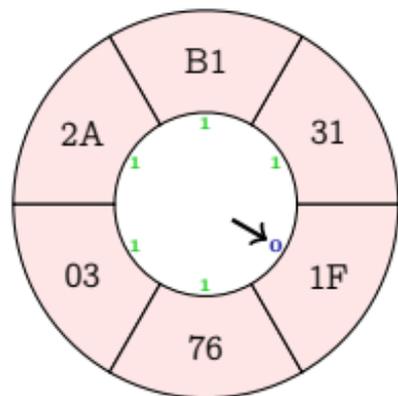
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

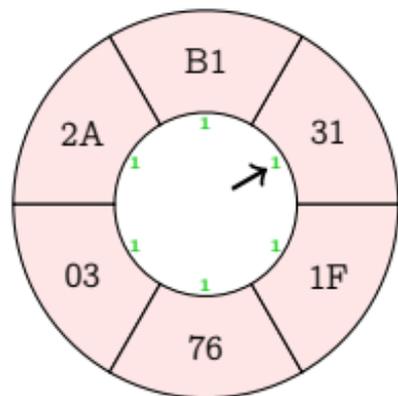
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

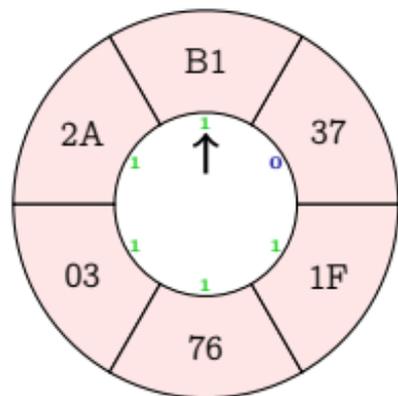
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Principe

- Bit de **2nd chance** mis à 0 lors d'un accès
- Lorsqu'on a besoin de libérer un cadre:
  - Si **bit=0**, mettre le bit à 1 et **passer**
  - Sinon **utiliser le cadre** (et remettre le bit à 0)

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



# Algorithmes basés sur l'utilisation

### Retirer la page la plus pertinente

- La page la moins utilisée (LFU)
- Une page peu utilisée (NRU)
- La page la moins récemment utilisée (LRU)

### Retirer la page la plus pertinente

- La page la moins utilisée (LFU)
- Une page peu utilisée (NRU)
- La page la moins récemment utilisée (LRU)

Avantage : moins de défaut de page

*En pratique, FIFO-2 fait 15 à 20 % plus de défauts que LRU*

Retirer la page la plus pertinente

- La page la moins utilisée (LFU)
- Une page peu utilisée (NRU)
- La page la moins récemment utilisée (LRU)

Avantage : moins de défaut de page

*En pratique, FIFO-2 fait 15 à 20 % plus de défauts que LRU*

Inconvénient : algorithmes plus complexes

- Espace mémoire supplémentaire
- Temps de calcul à chaque appel de page

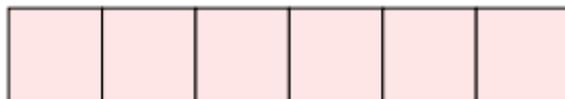
## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Page	Usage	Date
03	0	
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Page	Usage	Date
03	0	
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

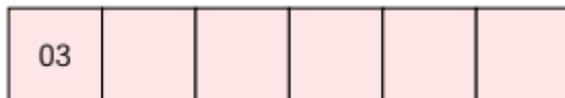
## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Page	Usage	Date
03	1	1
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04				
----	----	--	--	--	--

Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A			
----	----	----	--	--	--

Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	0	
2A	1	3
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	1	4
2A	1	3
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	1	4
2A	2	3
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	
----	----	----	----	----	--

Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	0	
37	0	
48	0	
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	48
----	----	----	----	----	----

Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	0	
37	0	
48	1	7
76	0	
B1	0	

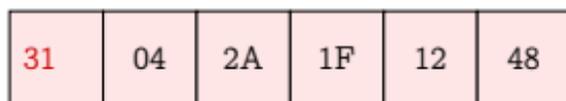
## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Usage = 1, Date = min(1,2,4,6,7)

Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	1	8
37	0	
48	1	7
76	0	
B1	0	

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	1	8
37	0	
48	1	7
76	0	
B1	1	9

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	0	
B1	1	9

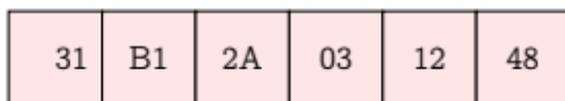
## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Usage = 1, Date = min(8,9,4,6,7)

Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	0	
B1	1	9

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	4	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	1F
----	----	----	----	----	----

Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	2	14
2A	4	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

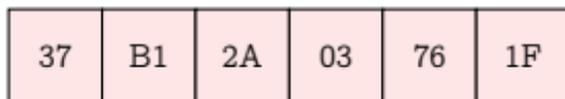
## Principe

Noter le taux d'utilisation de chaque page du processus.

- choisir la page la moins utilisée  
en cas d'égalité: FIFO

## Exemple:

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



Usage = 1, Date = min(8,9,12)

Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	2	14
2A	4	3
31	1	8
37	1	15
48	1	7
76	1	12
B1	1	9

### Nombre total d'utilisation

- ✗ **Problème:** une page beaucoup utilisée reste
- ✓ **Solution:** remettre usage à 0 périodiquement  
*fenêtre glissante trop coûteux à implémenter. . .*

## Nombre total d'utilisation

- ✗ **Problème:** une page beaucoup utilisée reste
- ✓ **Solution:** remettre usage à 0 périodiquement  
*fenêtre glissante trop coûteux à implémenter. . .*

## Performances

- ✓ Beaucoup moins de défaut de page que les autres

## Nombre total d'utilisation

- ✗ **Problème:** une page beaucoup utilisée reste
  - ✓ **Solution:** remettre usage à 0 périodiquement
- fenêtre glissante trop coûteux à implémenter. . .*

## Performances

- ✓ Beaucoup moins de défaut de page que les autres

## Mais aussi

- ✗ 64 bits de plus dans chaque ligne
- ✗  $\mathcal{O}(n)$  à chaque page manquante

## Observations

- Compter les utilisations est coûteux

## Observations

- Compter les utilisations est coûteux  
→ *FIFO-2* = 1 seul bit : mémoire + temps!

## Observations

- Compter les utilisations est coûteux  
→ *FIFO-2* = 1 seul bit : mémoire + temps!
- Certaines pages deviennent inutiles

## Observations

- Compter les utilisations est coûteux  
→ *FIFO-2* = 1 seul bit : mémoire + temps!
- Certaines pages deviennent inutiles  
→ remettre à 0 périodiquement

## Observations

- Compter les utilisations est coûteux  
→ *FIFO-2* = 1 seul bit : mémoire + temps!
- Certaines pages deviennent inutiles  
→ remettre à 0 périodiquement
- Les pages dans lesquelles on écrit sont plus susceptibles d'être réutilisées  
*un peu plus tard, en général. . .*

## Observations

- Compter les utilisations est coûteux  
→ *FIFO-2* = 1 seul bit : mémoire + temps!
- Certaines pages deviennent inutiles  
→ remettre à 0 périodiquement
- Les pages dans lesquelles on écrit sont plus susceptibles d'être réutilisées  
*un peu plus tard, en général. . .*  
→ Ajouter un 2e bit pour les pages modifiées

## Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

--	--	--	--	--	--

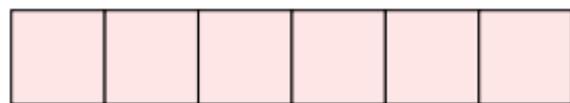
Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

## Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>



Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03					
----	--	--	--	--	--

R=1  
M=0

Page	R	M	Date
03	1	0	1
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03	04				
----	----	--	--	--	--

R=1 M=0      R=1 M=1

Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

03	04	2A			
----	----	----	--	--	--

R=1 M=0    R=1 M=1    R=1 M=0

Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	0	0	
2A	1	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

03	04	2A	1F		
----	----	----	----	--	--

R=1 M=0   R=1 M=1   R=1 M=0   R=1 M=1

Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	1	1	4
2A	1	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03	04	2A	1F		
----	----	----	----	--	--

R=1 M=0   R=1 M=1   R=1 M=0   R=1 M=1

RESET

Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	1	1	4
2A	1	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03	04	2A	1F		
----	----	----	----	--	--

R=0 M=0   R=0 M=1   R=0 M=0   R=0 M=1

RESET

Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	
1F	0	1	4
2A	0	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

03	04	2A	1F		
----	----	----	----	--	--

R=0 M=0    R=0 M=1    R=1 M=1    R=0 M=1

Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03	04	2A	1F	12	
----	----	----	----	----	--

R=0 R=0 R=1 R=0 R=1  
M=0 M=1 M=1 M=1 M=0

Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

03	04	2A	1F	12	48
R=0 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=0

Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	1	0	7
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

31	04	2A	1F	12	48
R=1 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=0

Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	1	0	8
37	0	0	
48	1	0	7
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

31	04	2A	1F	12	48
R=1 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=0

RESET

Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	1	0	8
37	0	0	
48	1	0	7
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

31	04	2A	1F	12	48
R=0 M=0	R=0 M=1	R=0 M=1	R=0 M=1	R=0 M=0	R=0 M=0

RESET

Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	6
1F	0	1	4
2A	0	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	0	0	

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

31	04	2A	1F	B1	48
R=0 M=0	R=0 M=1	R=0 M=1	R=0 M=1	R=1 M=0	R=0 M=0

Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	6
1F	0	1	4
2A	0	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	1	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow$  R=1 ; écriture  $\rightarrow$  M=1
- Priorité: (*FIFO en cas d'égalité*)  
(R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

31	04	2A	1F	B1	48
R=0 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=0 M=0

Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	1	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

31	04	2A	1F	B1	03
R=0 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=1

Page	R	M	Date
03	1	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	1	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

76	04	2A	1F	B1	03
R=1 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=1

Page	R	M	Date
03	1	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	1	0	12
B1	1	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

76	04	2A	1F	B1	03
R=1 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=1 M=0	R=1 M=1

RESET

Page	R	M	Date
03	1	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	1	0	12
B1	1	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

76	04	2A	1F	B1	03
R=0 M=0	R=0 M=1	R=0 M=1	R=0 M=1	R=0 M=0	R=0 M=1

RESET

Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	0	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03<sub>r</sub> 04<sub>w</sub> 2A<sub>r</sub> 1F<sub>w</sub> 2A<sub>w</sub> 12<sub>r</sub> 48<sub>r</sub> 31<sub>r</sub> B1<sub>r</sub> 2A<sub>r</sub> 03<sub>w</sub> 76<sub>r</sub> 2A<sub>w</sub> 1F<sub>w</sub> 37<sub>r</sub>

76	04	2A	1F	B1	03
R=0 M=0	R=0 M=1	R=1 M=1	R=0 M=1	R=0 M=0	R=0 M=1

Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow$  R=1 ; écriture  $\rightarrow$  M=1
- Priorité: (*FIFO en cas d'égalité*)  
(R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

76	04	2A	1F	B1	03
R=0 M=0	R=0 M=1	R=1 M=1	R=1 M=1	R=0 M=0	R=0 M=1

Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	1	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

# Not Recently Used

2 bits: R (page **référéncée**) et M (page **modifiée**)

- Lecture ou écriture  $\rightarrow R=1$  ; écriture  $\rightarrow M=1$
- Priorité: (*FIFO en cas d'égalité*)  
 $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- Tous les R sont remis à 0 **chaque K cycles**

03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

76	04	2A	1F	37	03
R=0 M=0	R=0 M=1	R=1 M=1	R=1 M=1	R=1 M=0	R=0 M=1

Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	1	1	4
2A	1	1	3
31	0	0	8
37	1	0	15
48	0	0	7
76	0	0	12
B1	0	0	9

## Performance

- ✓ Peu de défaut de page
- ✓ Peu coûteux en mémoire
- ✗ Temps de calcul:  $\mathcal{O}(n)$  à chaque reset et page fault

## Performance

- ✓ Peu de défaut de page
- ✓ Peu coûteux en mémoire
- ✗ Temps de calcul:  $\mathcal{O}(n)$  à chaque reset et page fault

## Gain

En pratique, gain trop faible par rapport à FIFO-2

## Principe

- File (FIFO) avec remise en fin
- Implémentation matérielle  $\rightarrow$  calcul en  $\mathcal{O}(1)$

## Principe

- File (FIFO) avec remise en fin
- Implémentation matérielle  $\rightarrow$  calcul en  $\mathcal{O}(1)$

## Implémentation

- Matrice triangulaire  $N =$  nombre de cadres sans la diagonale, tout initialisé à 0

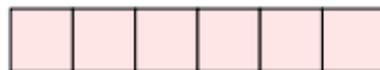
	0	1	2	3	4	5
0		0	0	0	0	0
1			0	0	0	0
2				0	0	0
3					0	0
4						0
5						

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	0	0
1		0	0	0	0	0
2			0	0	0	0
3				0	0	0
4					0	0
5						0

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

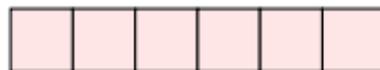


## Implémentation

- Matrice triangulaire  $N = \text{nombre de cadres}$
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	1		0	0	0	0
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

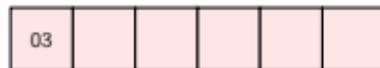


## Implémentation

- Matrice triangulaire  $N = \text{nombre de cadres}$
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	1	1	1	1	1
1	1		0	0	0	0
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

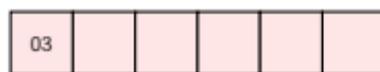


## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		1	1	1	1	1
1			0	0	0	0
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

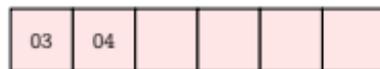


## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		1	1	1	1	1
1			0	0	0	0
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

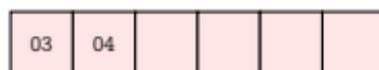


## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	1	1	1	1
1			1	1	1	1
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	1	1	1	1
1			1	1	1	1
2				0	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A			
----	----	----	--	--	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	1	1	1
1			0	1	1	1
2				1	1	1
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A			
----	----	----	--	--	--

## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	1	1	1
1			0	1	1	1
2				1	1	1
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	1	1
1			0	0	1	1
2				0	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	1	1
1			0	0	1	1
2				0	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	1	1
1			0	0	1	1
2				1	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F		
----	----	----	----	--	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre i  
→ ligne i à 1 puis colonne i à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne i)

	0	1	2	3	4	5
0		0	0	0	1	1
1			0	0	1	1
2				1	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	
----	----	----	----	----	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				1	0	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	
----	----	----	----	----	--

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				1	0	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre i  
→ ligne i à 1 puis colonne i à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne i)

	0	1	2	3	4	5
0	0	0	0	0	0	0
1		0	0	0	0	0
2			0	0	0	0
3				1	0	0
4					0	0
5						0

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

03	04	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	1	0	0	0	0	0
2			0	0	0	0
3				1	0	0
4					0	0
5						0

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	04	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		1	1	1	1	1
1			0	0	0	0
2				1	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	04	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		1	1	1	1	1
1	1		0	0	0	0
2				1	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	1	1	1	1
1			1	1	1	1
2				1	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	1	1	1	1
1			1	1	1	1
2				1	0	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	1	1	1
1			0	1	1	1
2				1	1	1
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	1F	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	1	1	1
1			0	1	1	1
2				1	1	1
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	1	1
1			0	0	1	1
2				0	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	12	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	1	1
1			0	0	1	1
2				0	1	1
3					1	1
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				0	0	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				0	0	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre i  
→ ligne i à 1 puis colonne i à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne i)

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				1	1	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	48
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		0	0	0	0	1
1			0	0	0	1
2				1	1	1
3					0	1
4						1
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	1F
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	0	0
1		0	0	0	0	0
2			0	0	0	0
3				1	1	0
4					0	0
5						0

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

31	B1	2A	03	76	1F
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire  $N =$  nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	1	0	0	0	0	0
2			0	0	0	0
3				1	1	0
4					0	0
5						0

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

37	B1	2A	03	76	1F
----	----	----	----	----	----

## Implémentation

- Matrice triangulaire N = nombre de **cadres**
- Utilisation d'une page dans le cadre  $i$   
→ ligne  $i$  à 1 puis colonne  $i$  à 0
- Cadre le plus ancien (à utiliser) =  
ligne est remplie de 0, colonne remplie de 1 (sauf ligne  $i$ )

	0	1	2	3	4	5
0		1	1	1	1	1
1			0	0	0	0
2				1	1	0
3					0	0
4						0
5						

03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

37	B1	2A	03	76	1F
----	----	----	----	----	----

### Ce qu'il faut retenir

- Politique de remplacement de page
- FIFO (avec bit de seconde chance)
- LRU (implémentée au niveau matériel)
- Tracer l'exécution et compter les défauts