Mémoire 3/3

Thomas Lavergne
lavergne@lisn.fr

Limites de la pagination

Avantages

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

Limites de la pagination

Avantages

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

Inconvénients

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

Limites de la pagination

Taille de page arbitraire

Impossible de vérifier finement les accès en dehors de la mémoire

Limites de la pagination

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

Limites de la pagination

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

Segmentation paginée

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

Segmentation paginée

1er niveau

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

- la traduction d'adresse permet d'obtenir une

adresse linéaire

2ème niveau

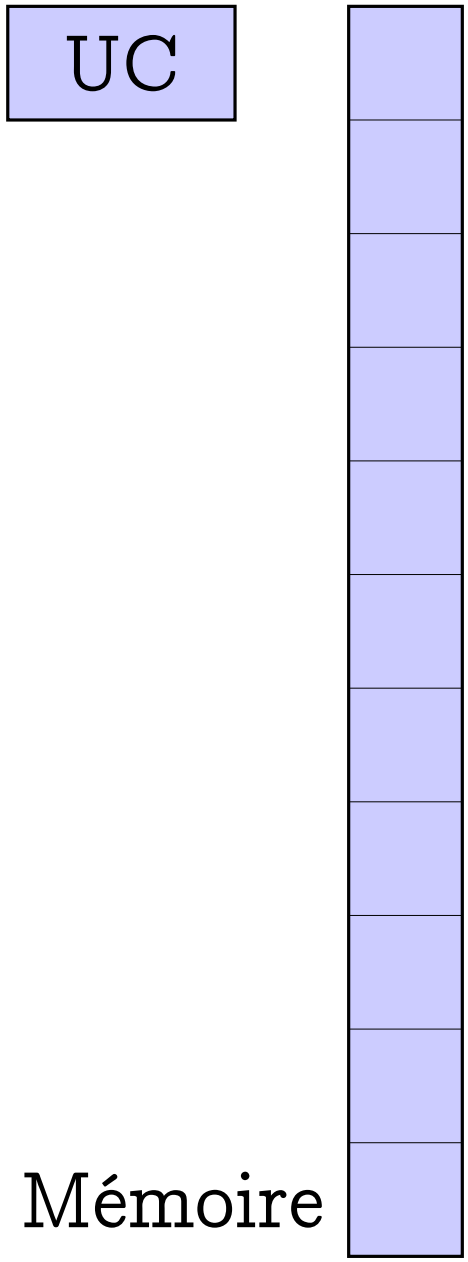
L'espace linéaire est géré de manière paginée.

- la traduction d'adresse permet d'obtenir une

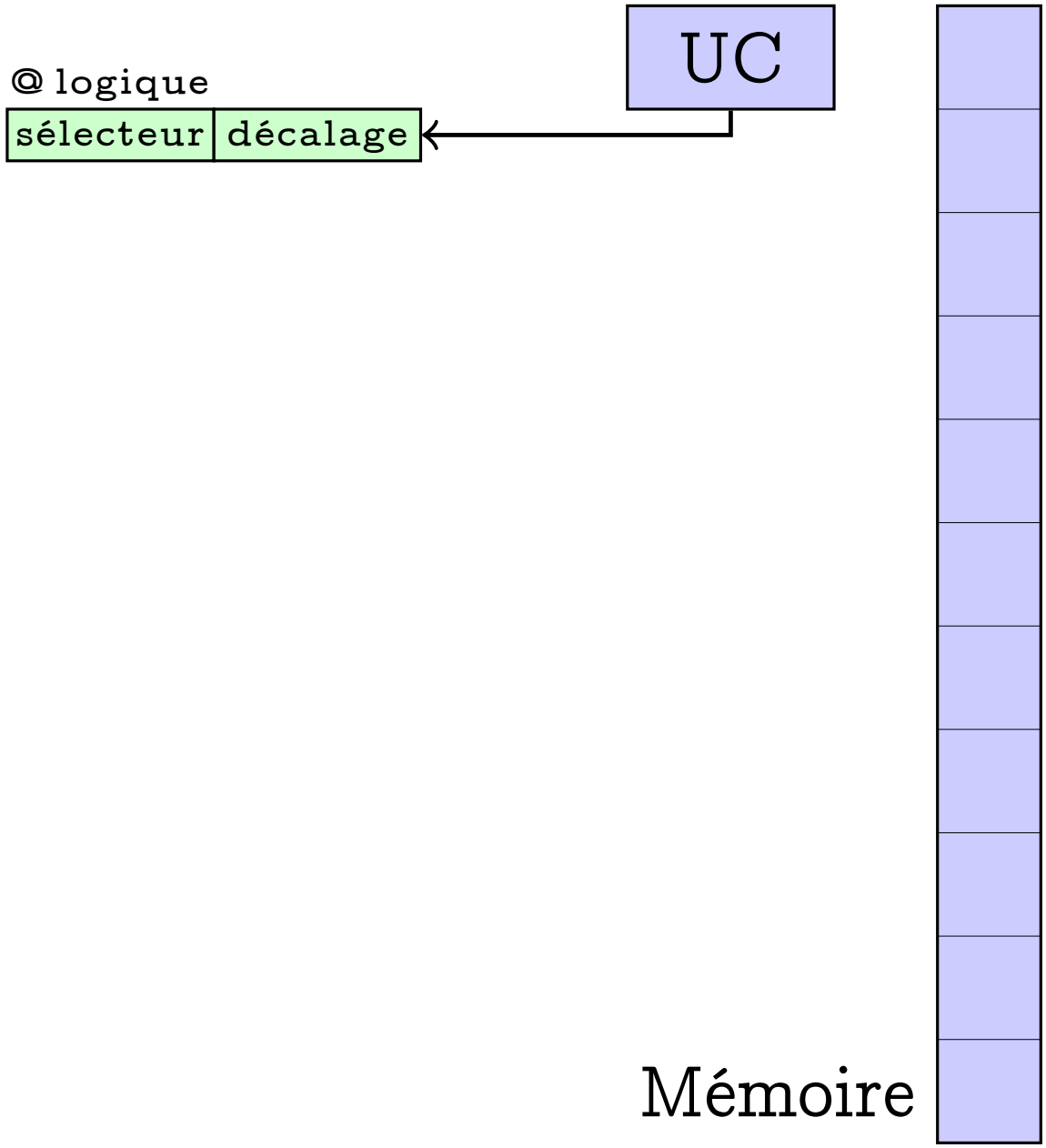
adresse physique

Le meilleur des deux mondes

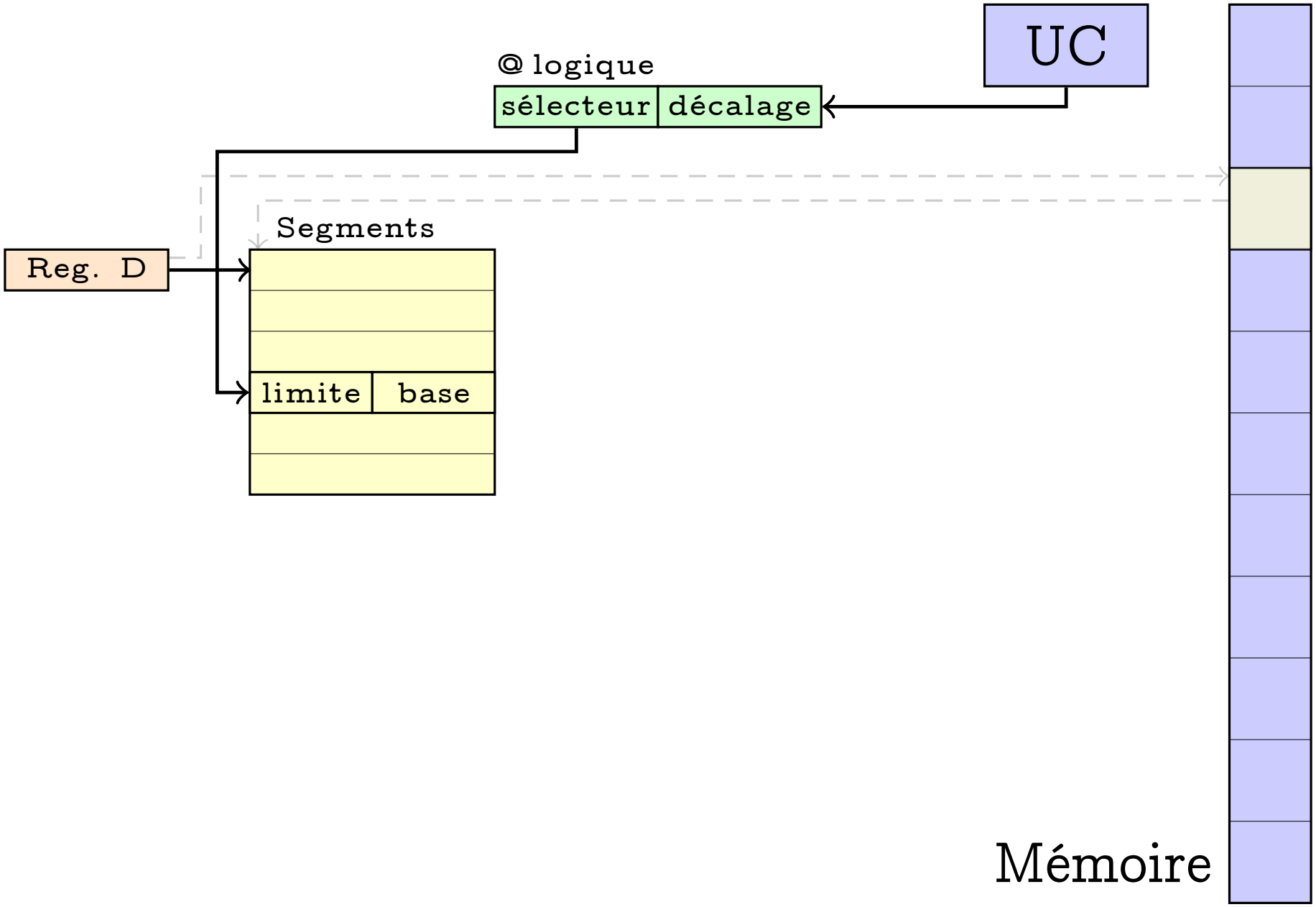
Segmentation et pagination à deux niveaux



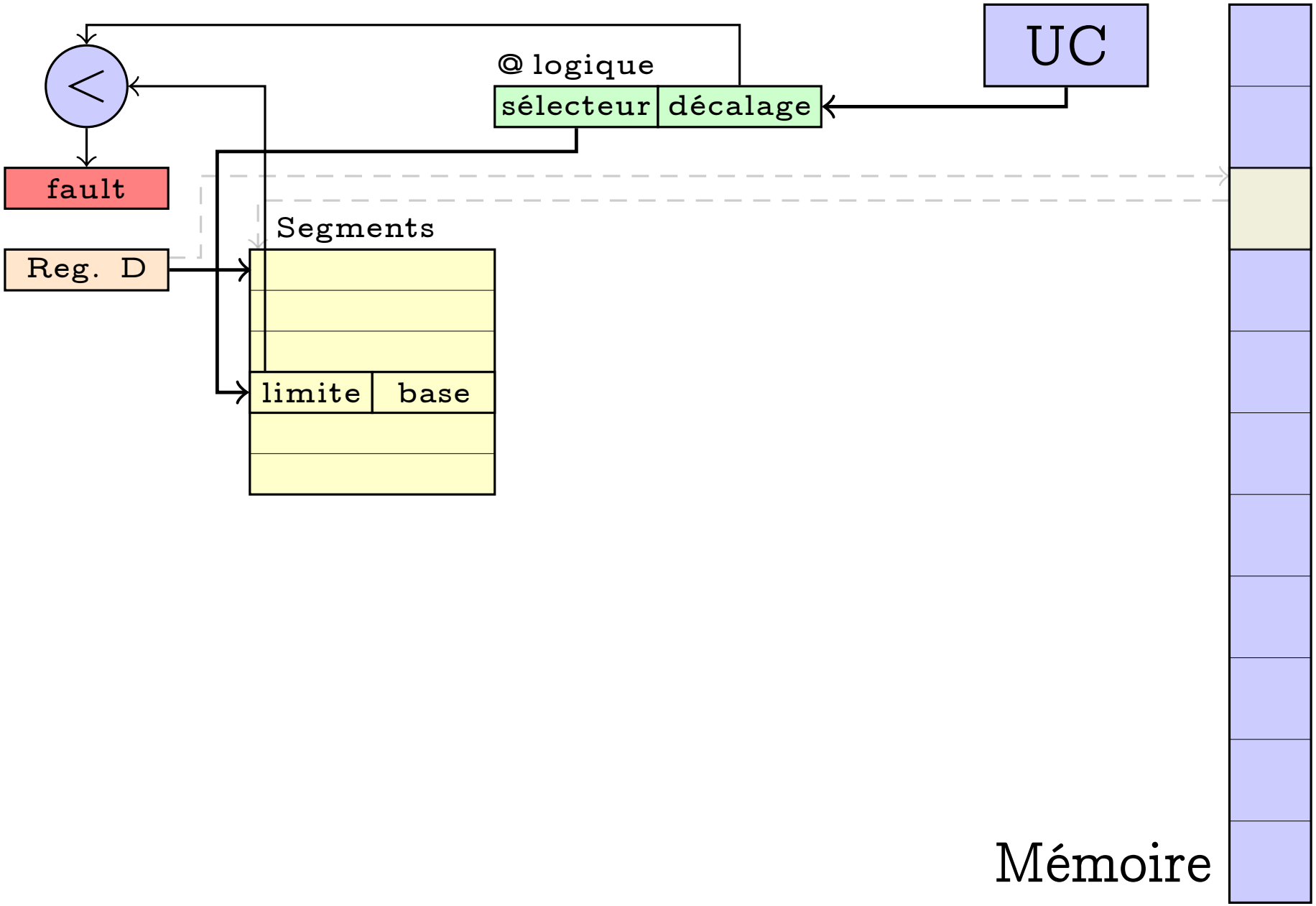
Segmentation et pagination à deux niveaux



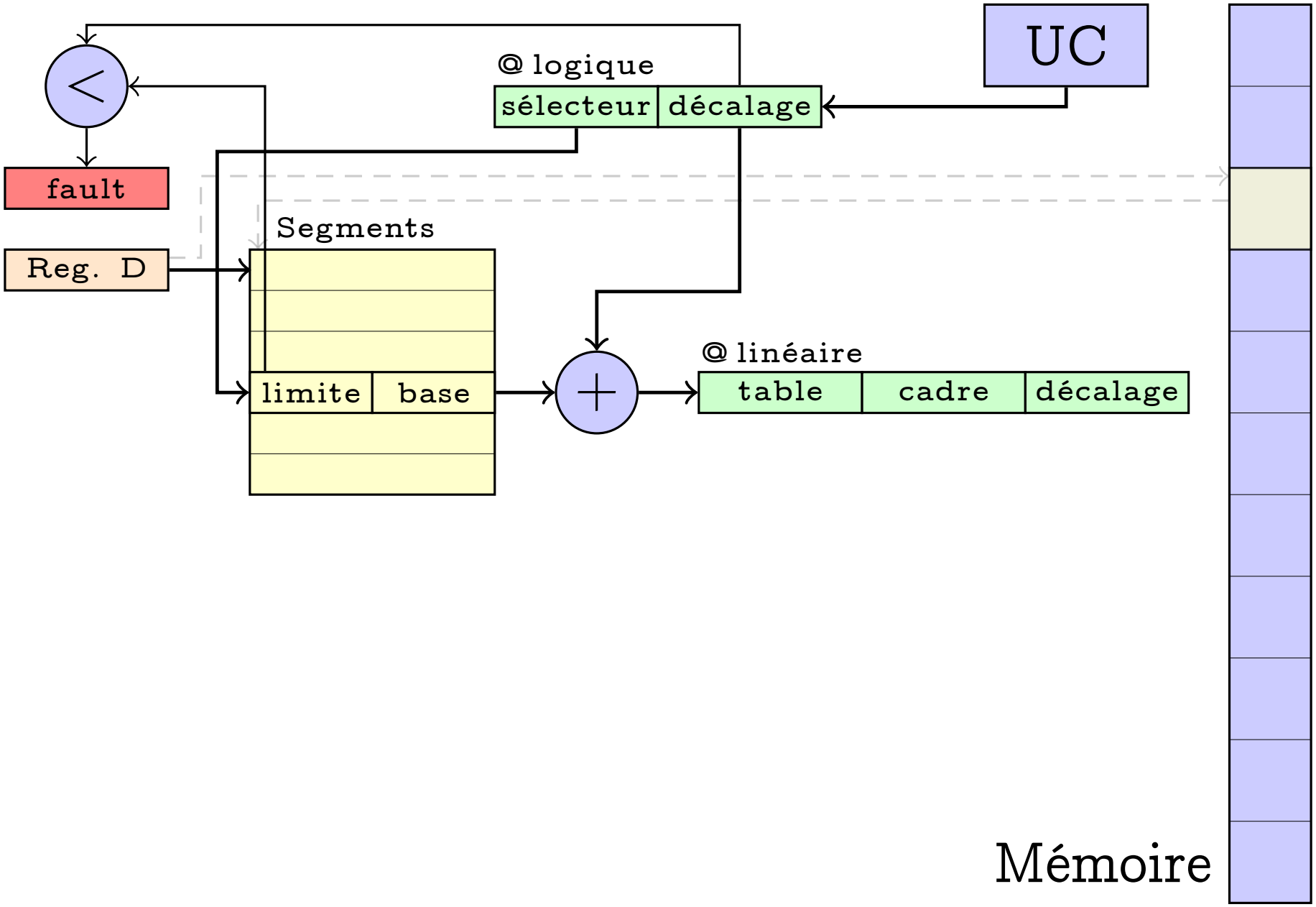
Segmentation et pagination à deux niveaux



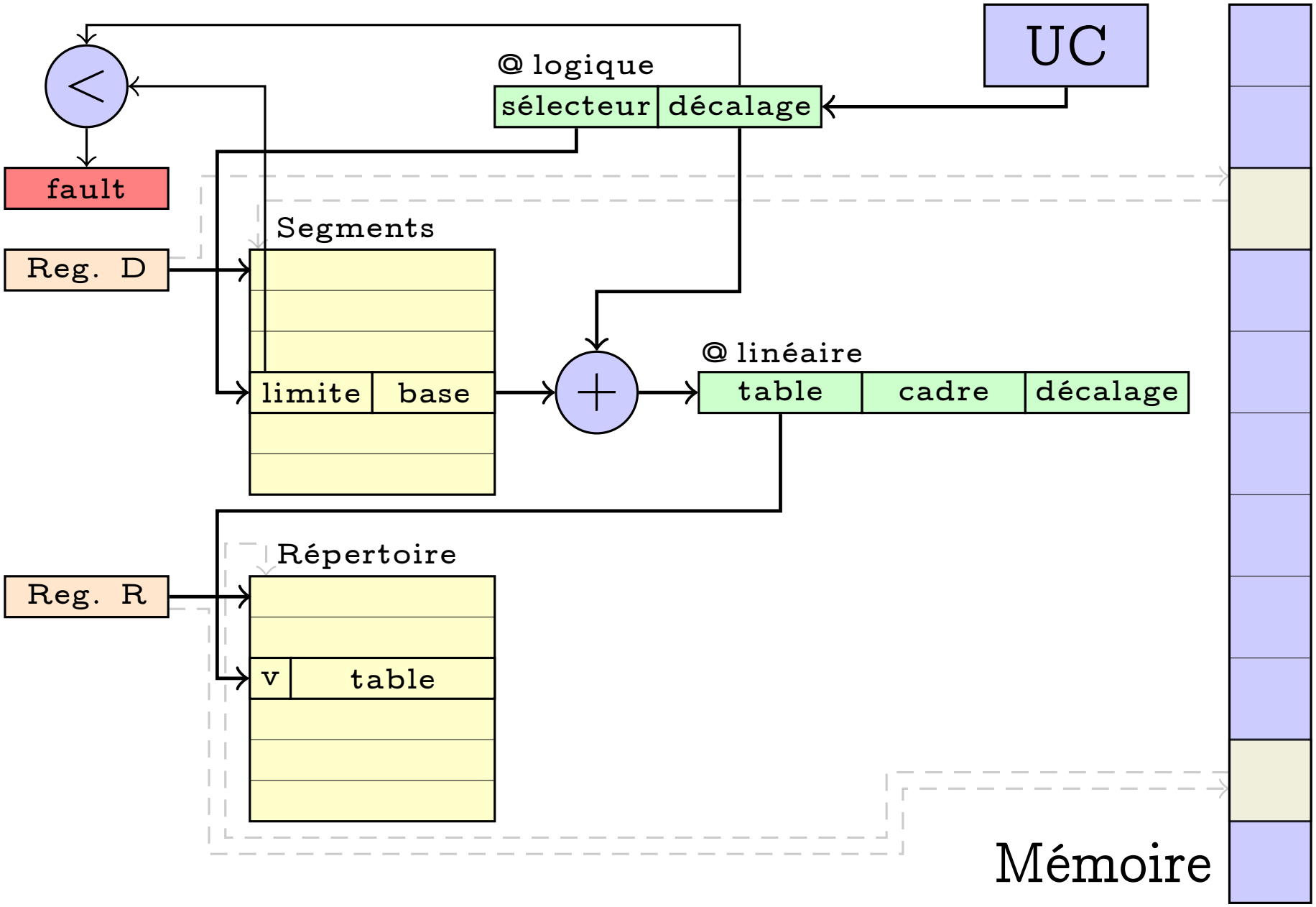
Segmentation et pagination à deux niveaux



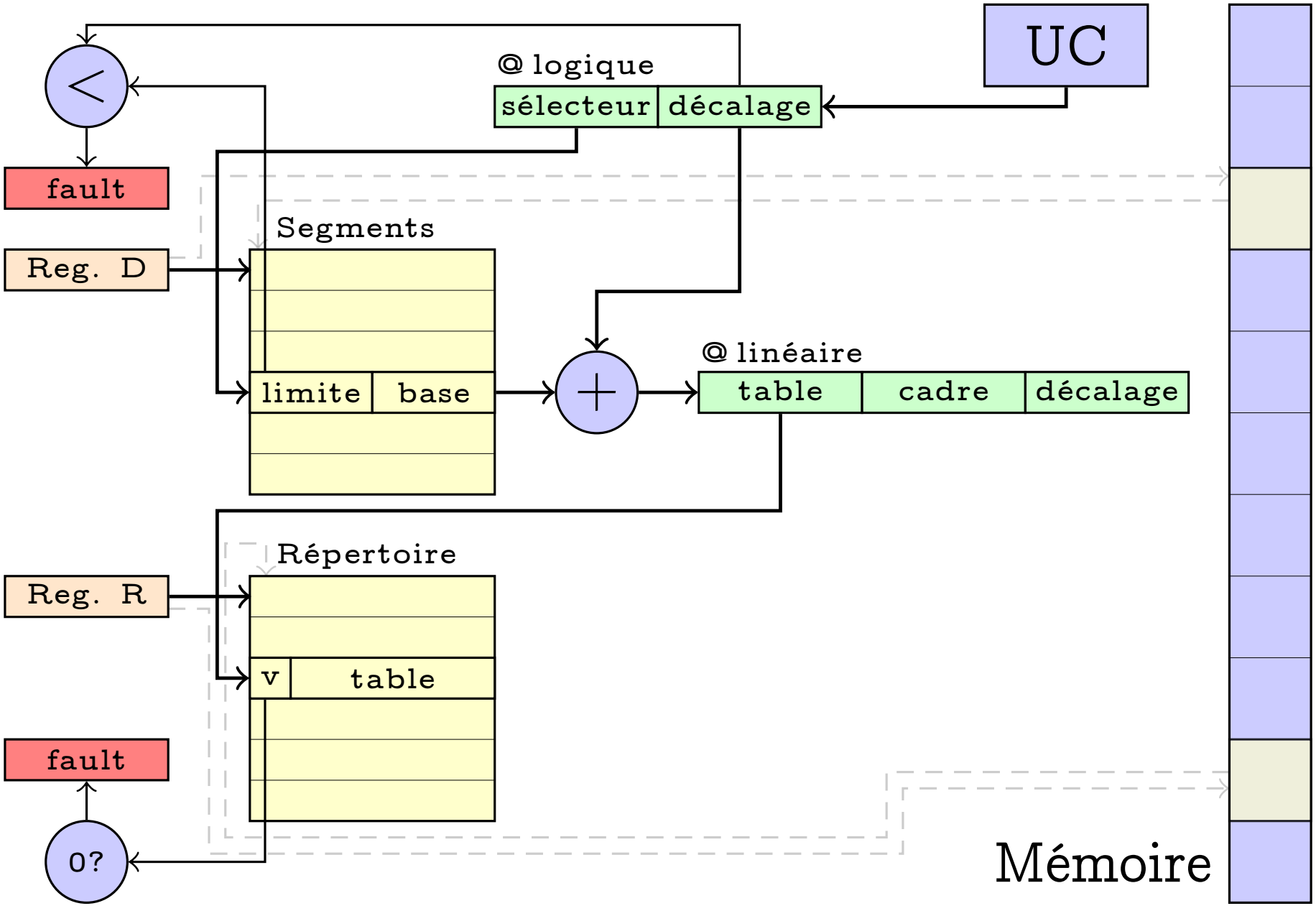
Segmentation et pagination à deux niveaux



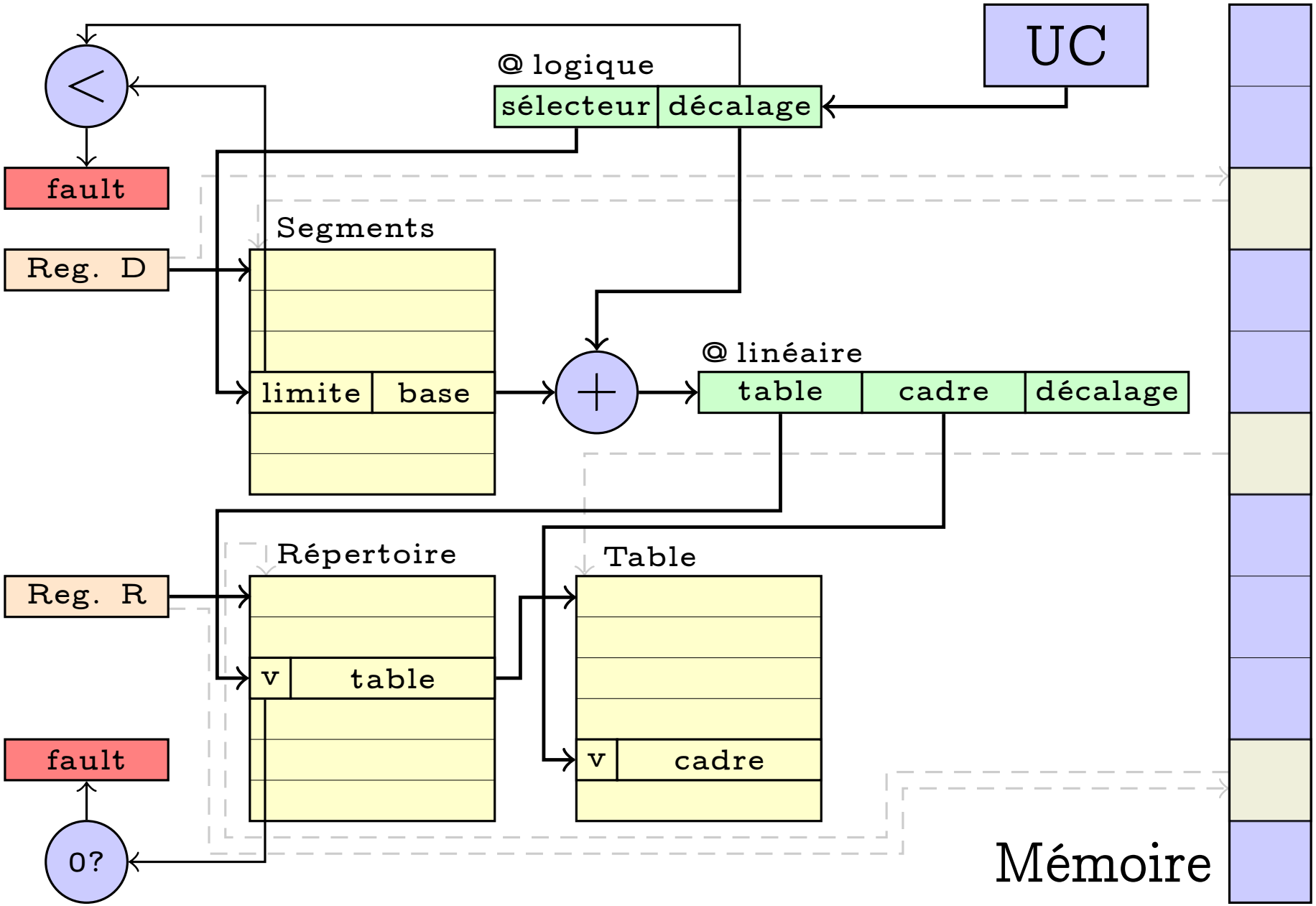
Segmentation et pagination à deux niveaux



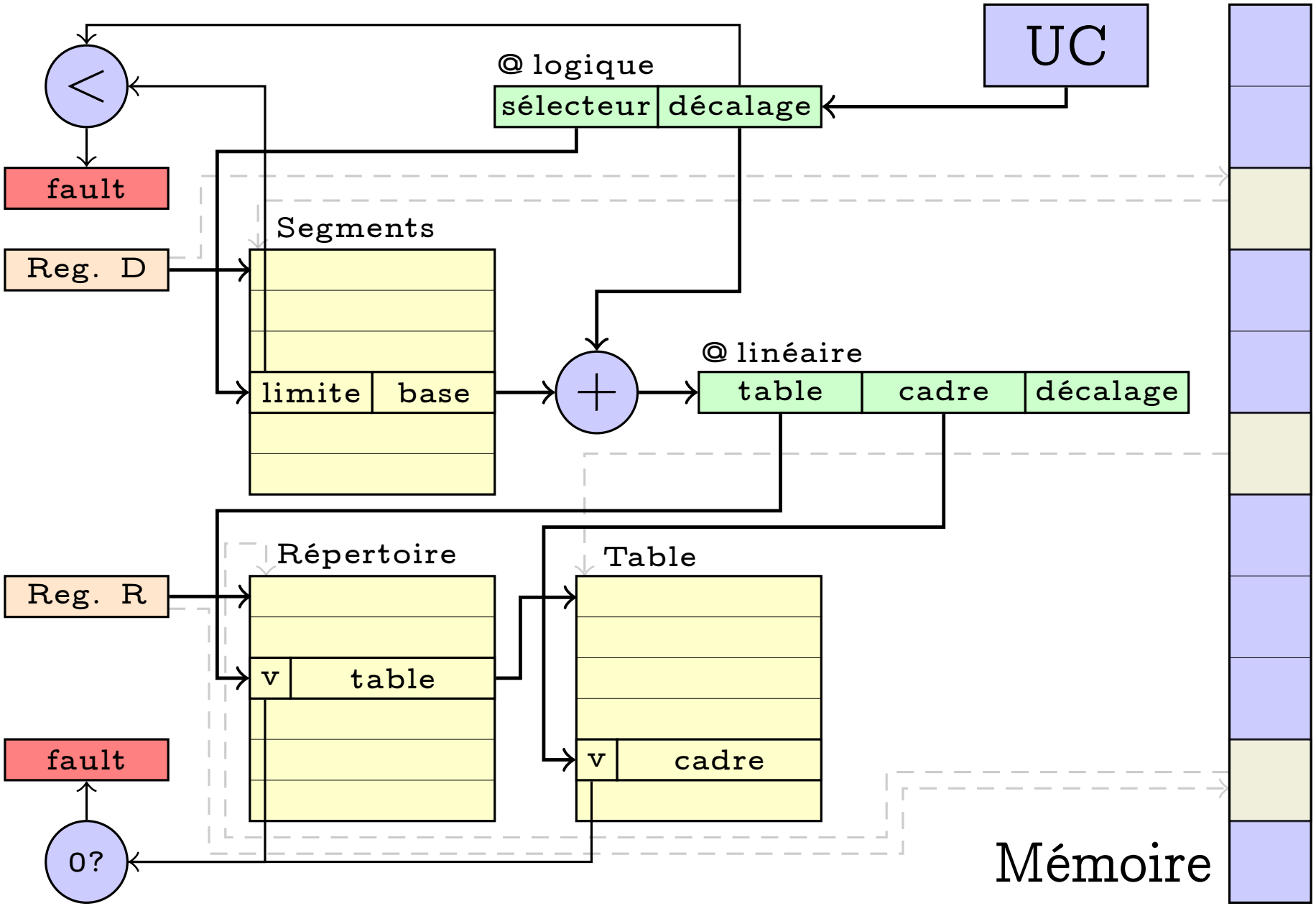
Segmentation et pagination à deux niveaux



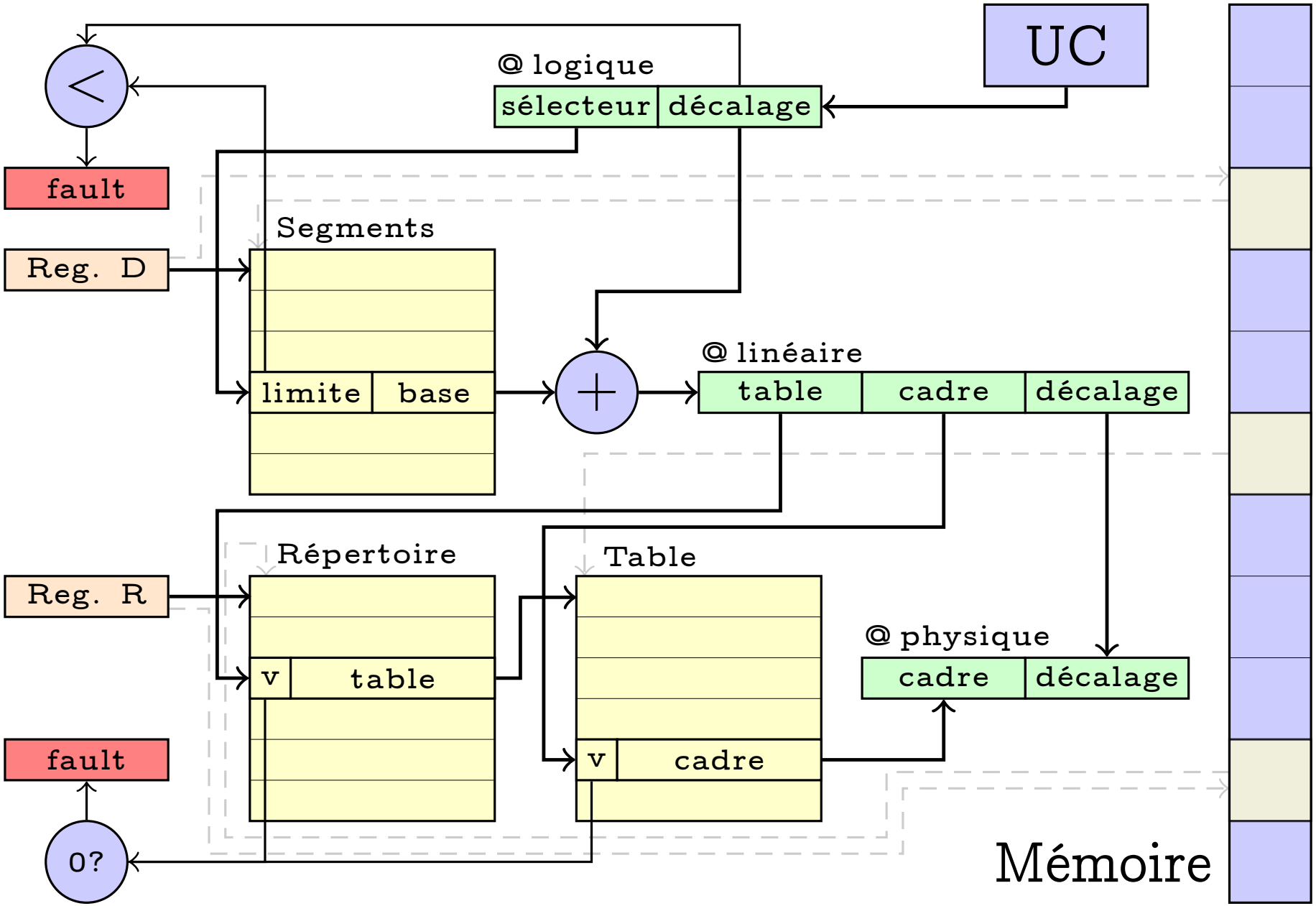
Segmentation et pagination à deux niveaux



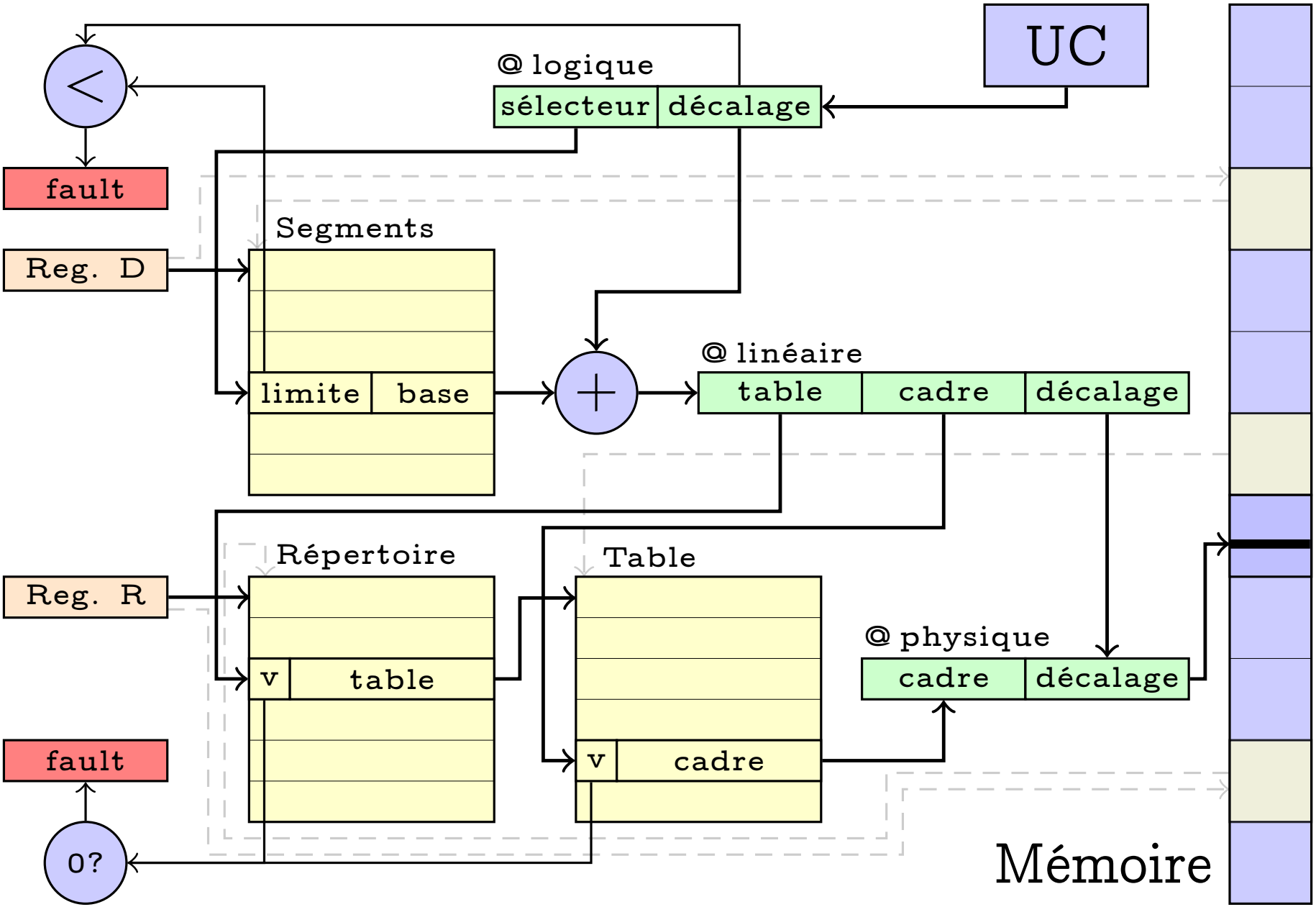
Segmentation et pagination à deux niveaux



Segmentation et pagination à deux niveaux



Segmentation et pagination à deux niveaux



Architecture x86

- Disponible depuis le 386
- Presque abandonné depuis le passage au 64bits

Architecture x86

- Disponible depuis le 386
- Presque abandonné depuis le passage au 64bits

Pourquoi ?

- X Quasiment unique donc peu portable
- X Lourd à gérer pour le système
- X Nécessite la coopération du compilateur
- X Trop peu de gain au final

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

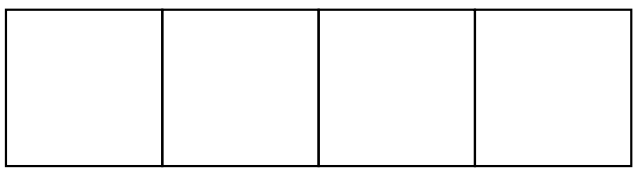
➜ Libérer un cadre lorsqu'on a besoin d'une nouvelle page

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04



Défauts : 0

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04

03	2A	1F	04

Défauts : 4

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04 2A

03	2A	1F	04

Défauts : 4

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04 2A

03	2A	1F	04

Défauts : 4

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04 2A 12

03	2A	1F	04

Défauts : 4

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

Exemple : 03 2A 1F 04 2A 12

12	2A	1F	04

Défauts : 5

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

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

12	2A	1F	04

Défauts : 5

Remplacement de pages

Mémoire virtuelle

Chaque processus dispose d'un nombre de cadres limité

selon la politique d'allocation

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

Problème (rappel)

Temps d'exécution proportionnel à la probabilité de défaut de page

➜ Réduire le nombre de défauts de page

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

03	2A	1F	04

Défauts : 6

Représentation visuelle

Objectif

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

Représentation visuelle

Objectif

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

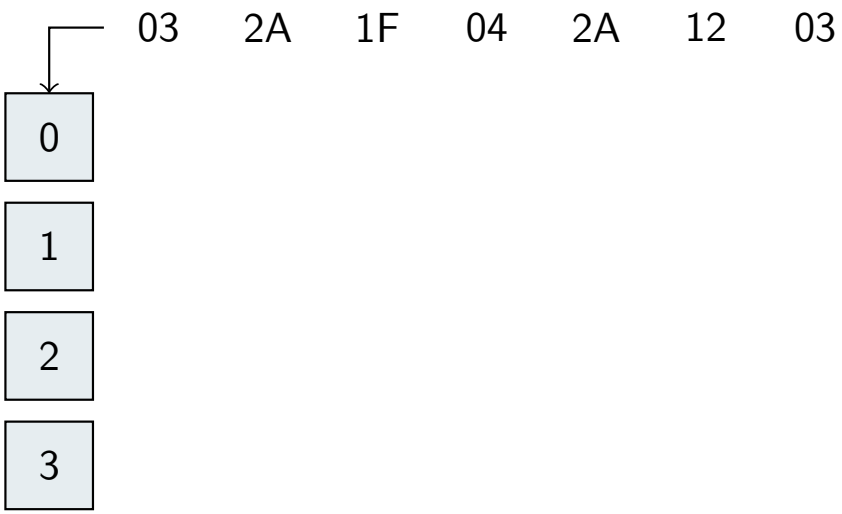
03 2A 1F 04 2A 12 03

Première ligne : liste des pages demandées (dans l'ordre)

Représentation visuelle

Objectif

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

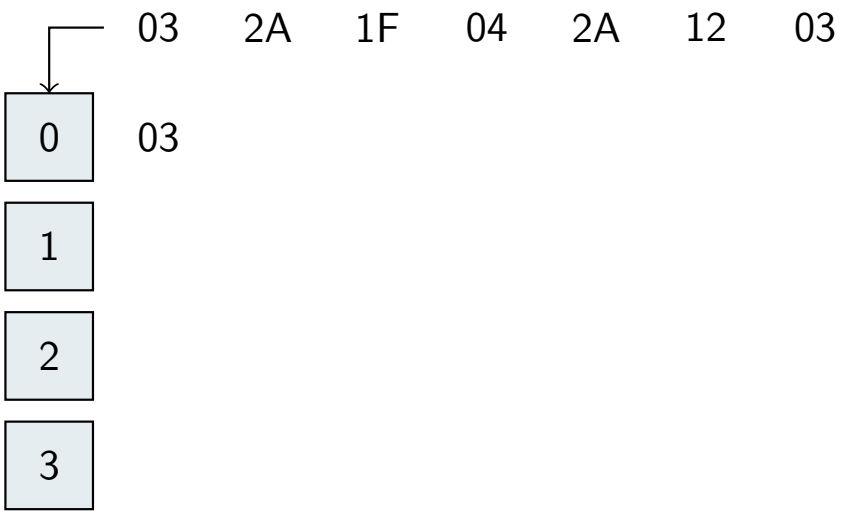


Une ligne par cadre de page alloué au processus

Représentation visuelle

Objectif

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

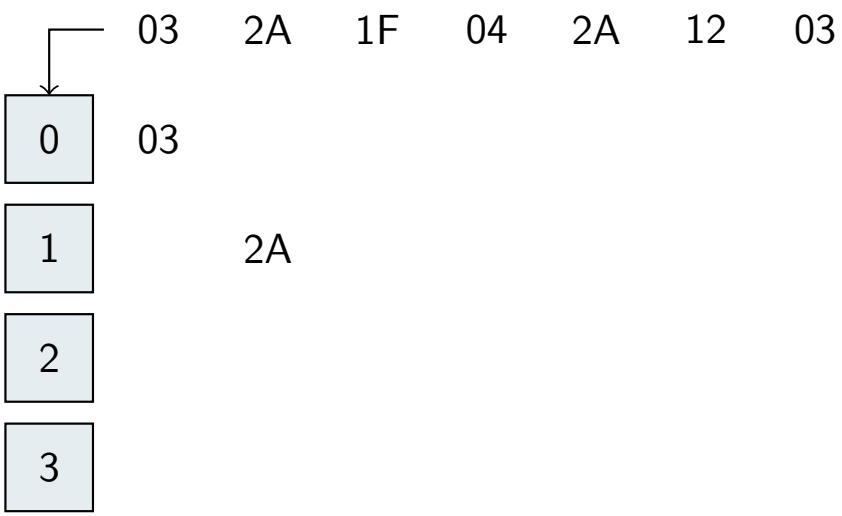


Une page par colonne

Représentation visuelle

Objectif

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

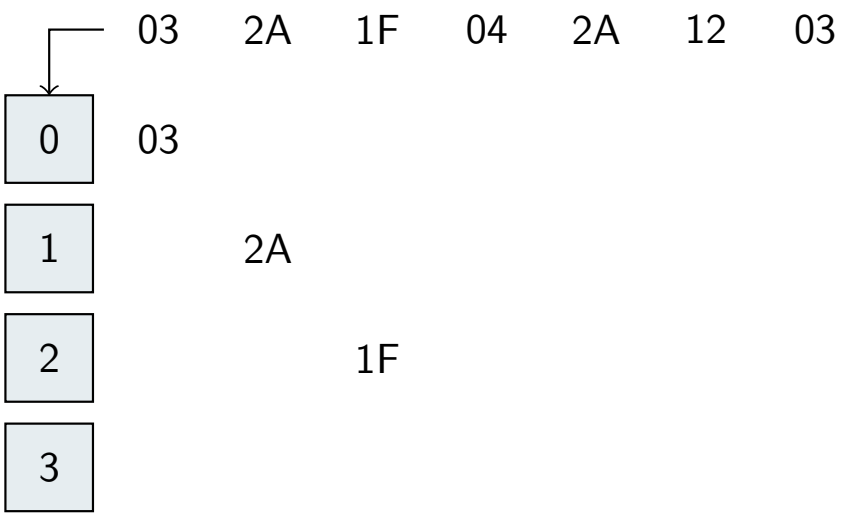


Une page par colonne

Représentation visuelle

Objectif

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

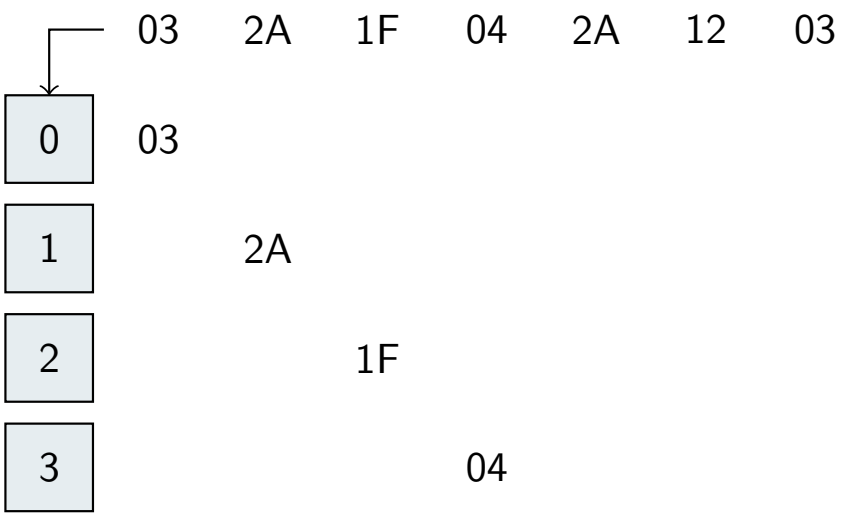


Une page par colonne

Représentation visuelle

Objectif

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

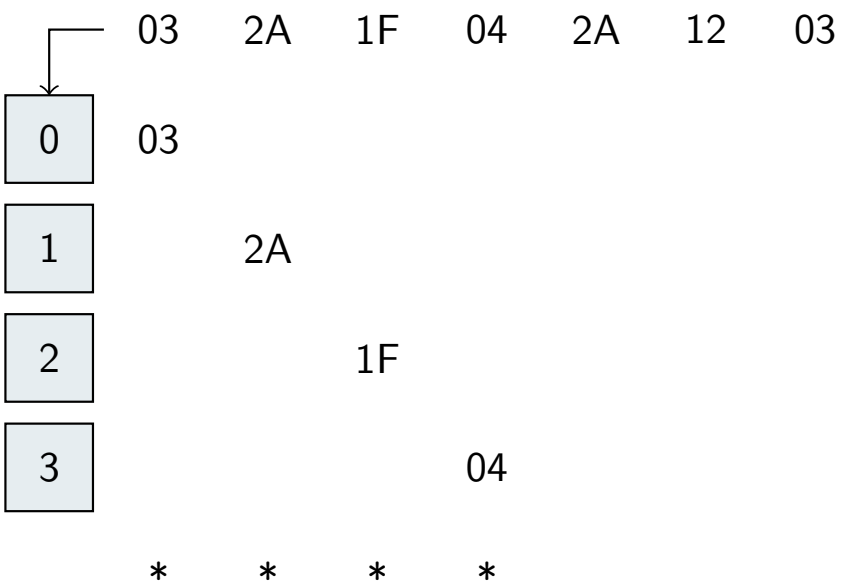


Une page par colonne

Représentation visuelle

Objectif

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

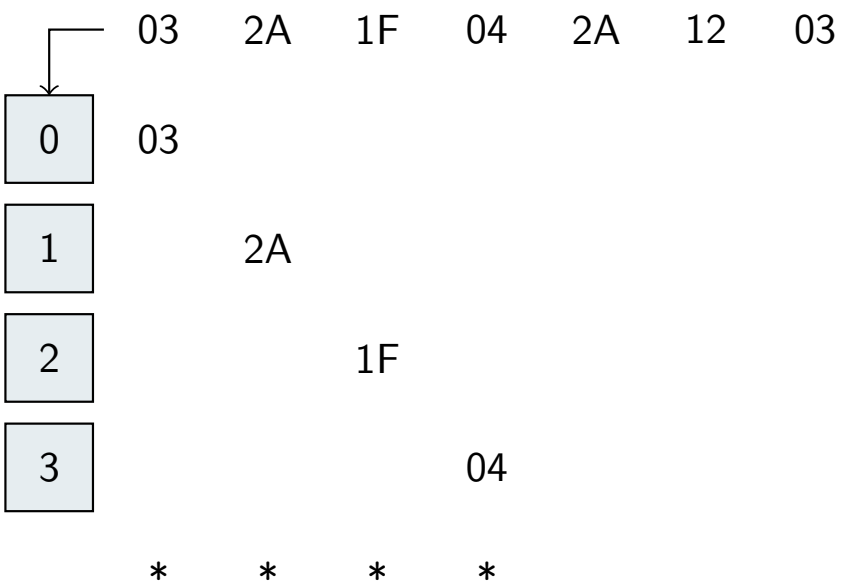


Marquer les défauts au fur et à mesure

Représentation visuelle

Objectif

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

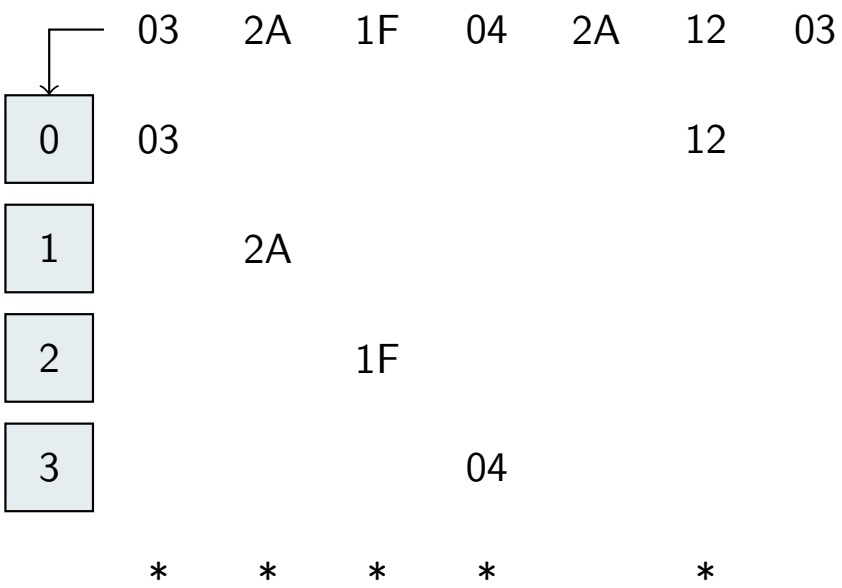


Pas de défaut → laisser en blanc

Représentation visuelle

Objectif

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

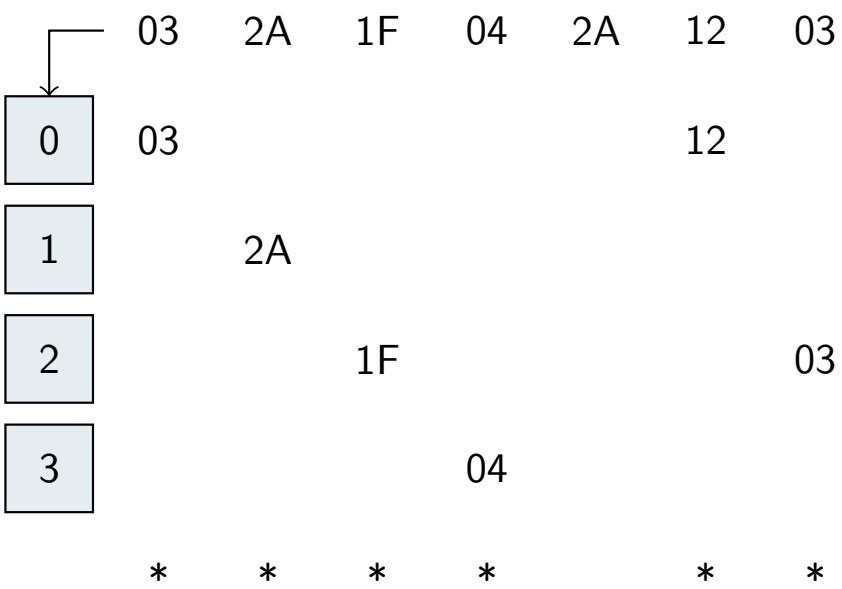


Et on continue...

Représentation visuelle

Objectif

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

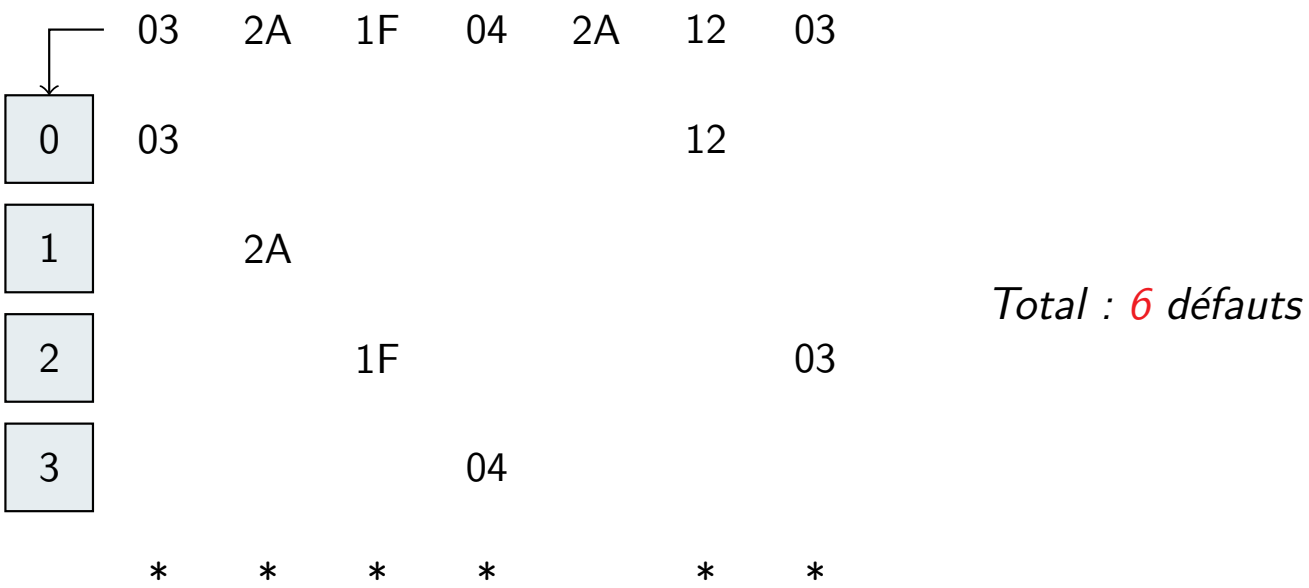


Et on continue...

Représentation visuelle

Objectif

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



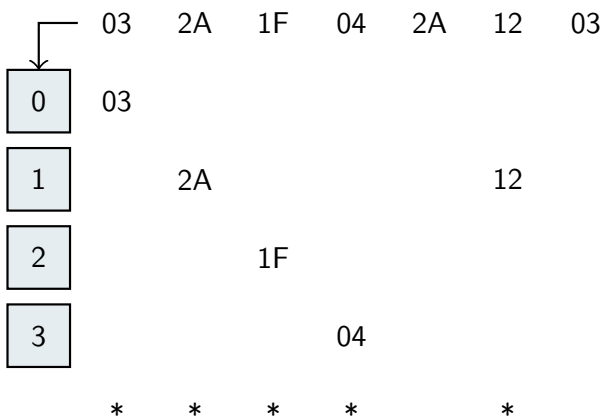
Total : 6 défauts

Et on continue. . .

Solution optimale

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

En pratique, on ne connaît pas les pages !

Solution

Algorithme de remplacement

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

Solution

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

Solution

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*

Principe

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

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

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

03	04				

Principe

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			

Principe

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		

Principe

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		

Principe

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	

Principe

File

Retirer la page la plus ancienne

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

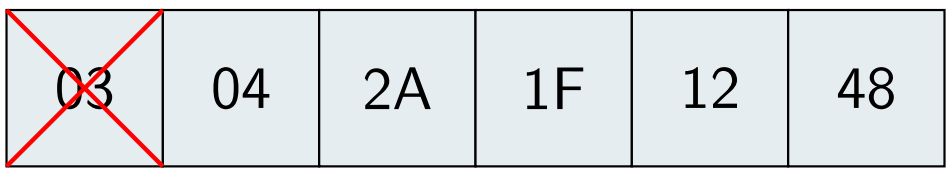
03	04	2A	1F	12	48

Principe

File

Retirer la page la plus ancienne

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



Principe

File

Retirer la page la plus ancienne

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

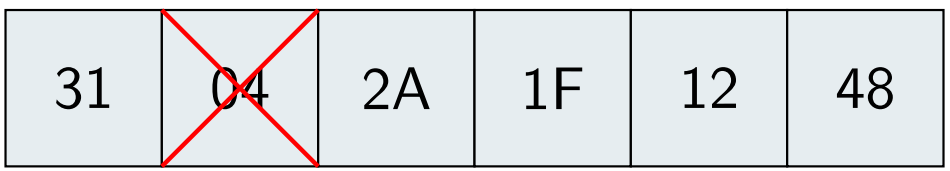
31	04	2A	1F	12	48

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

31	B1	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

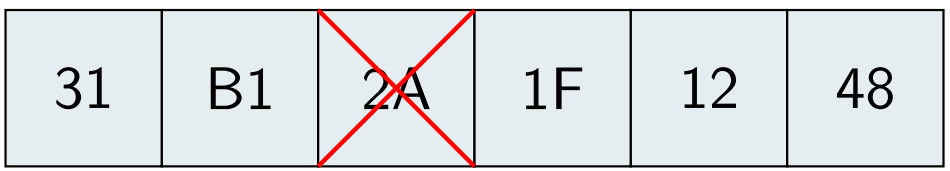
31	B1	2A	1F	12	48

Principe

File

Retirer la page la plus ancienne

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



Principe

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

Principe

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

Principe

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

Principe

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

Principe

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

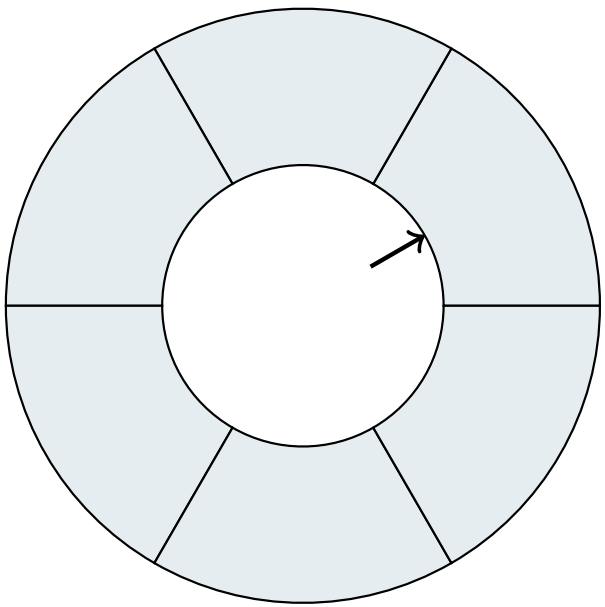
Principe

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



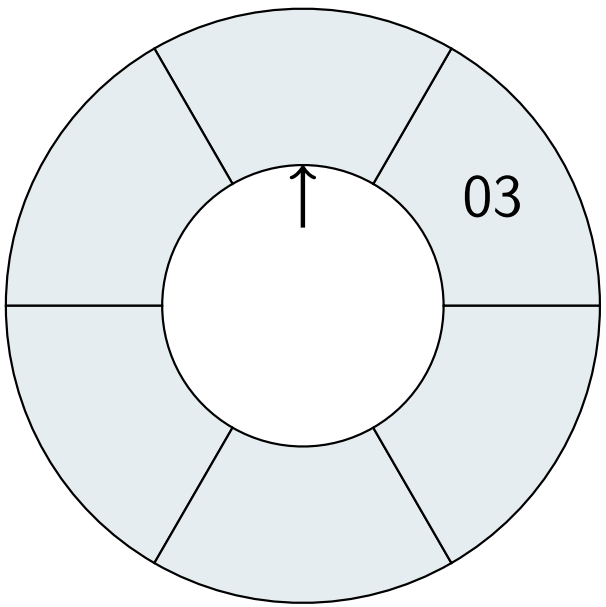
Principe

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



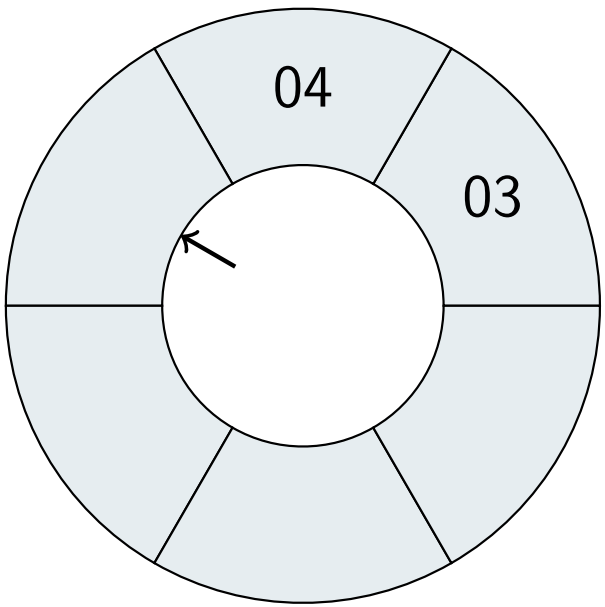
Principe

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



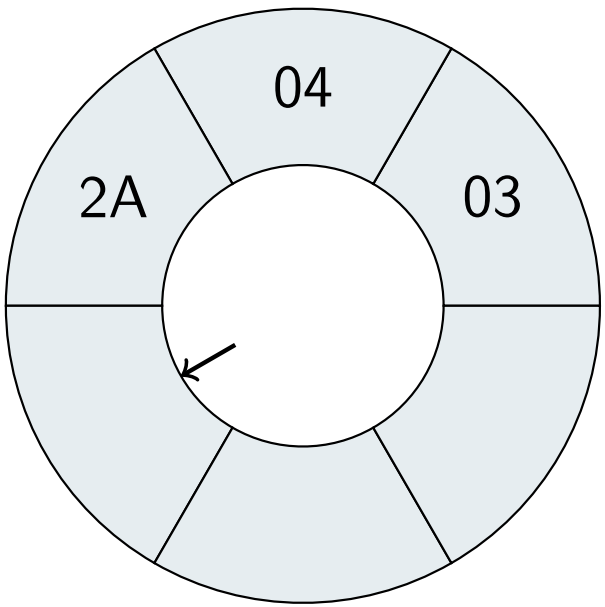
Principe

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



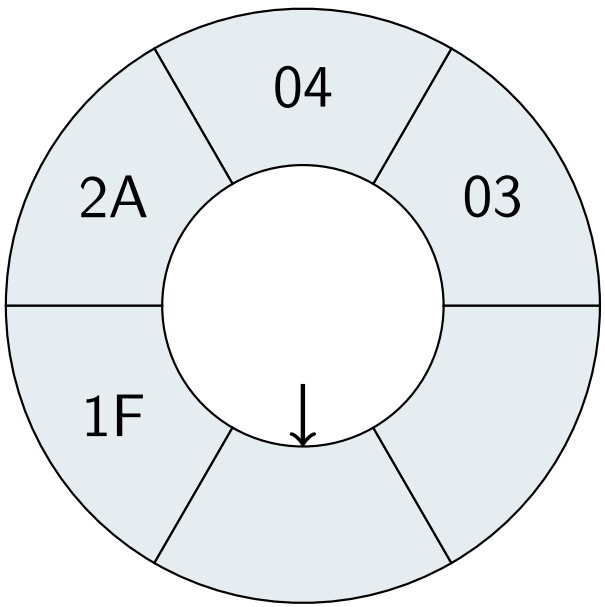
Principe

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



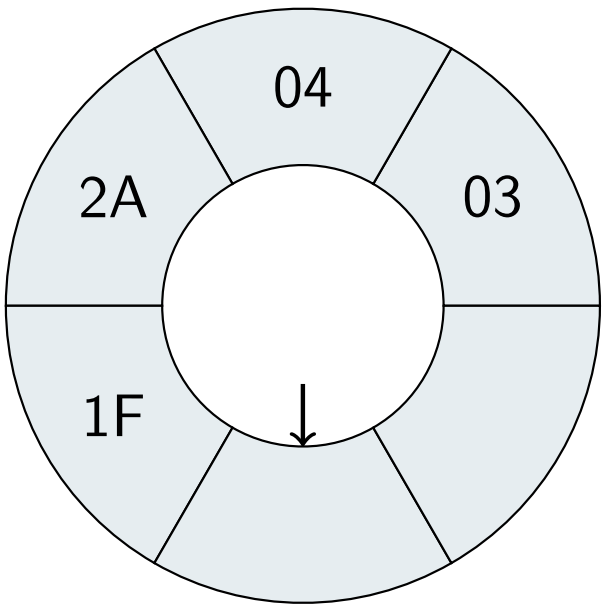
Principe

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



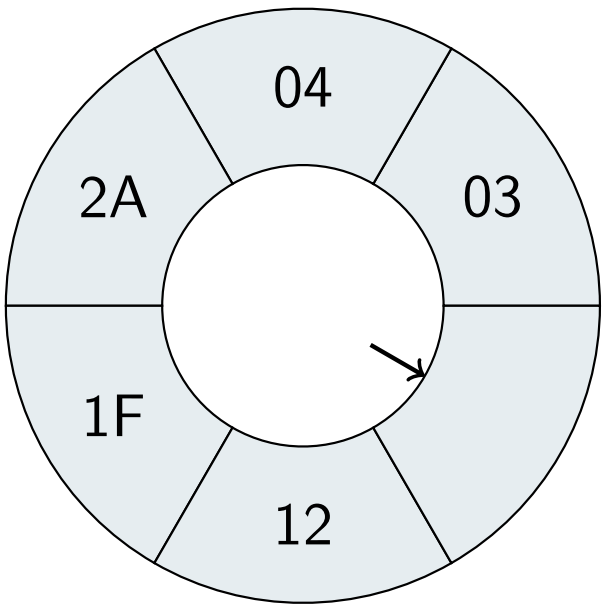
Principe

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



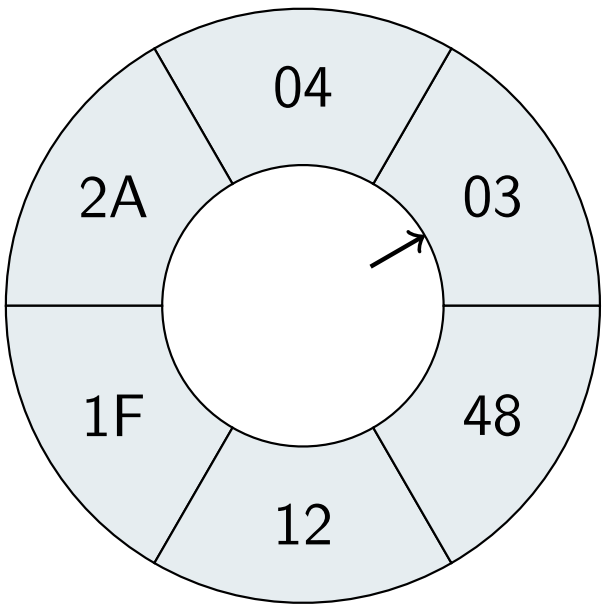
Principe

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



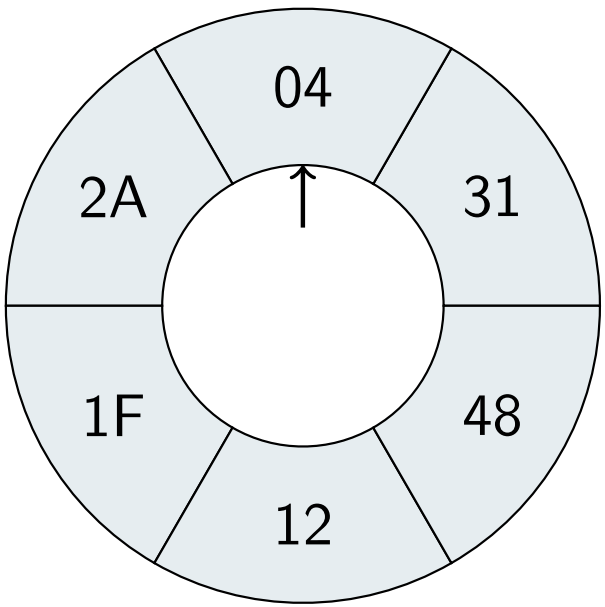
Principe

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



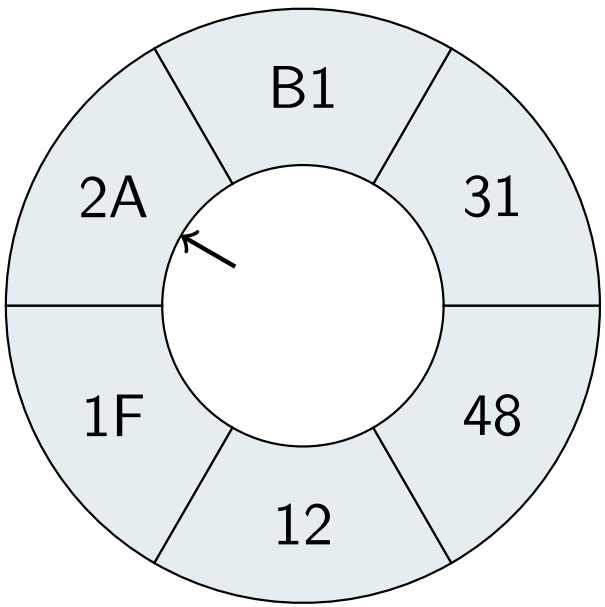
Principe

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



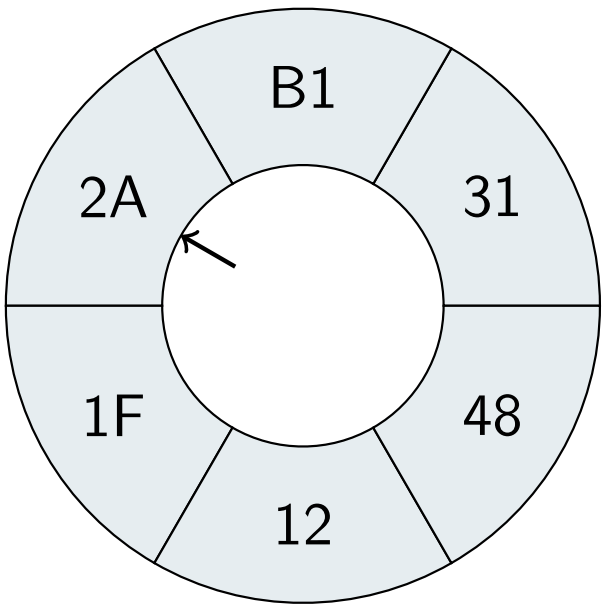
Principe

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



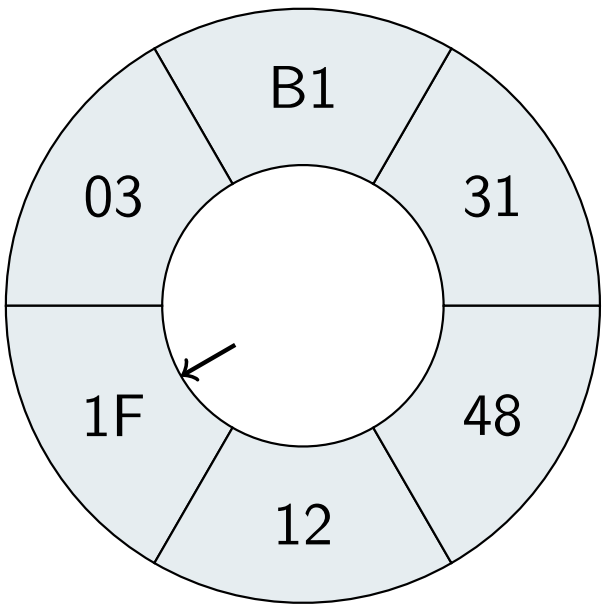
Principe

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



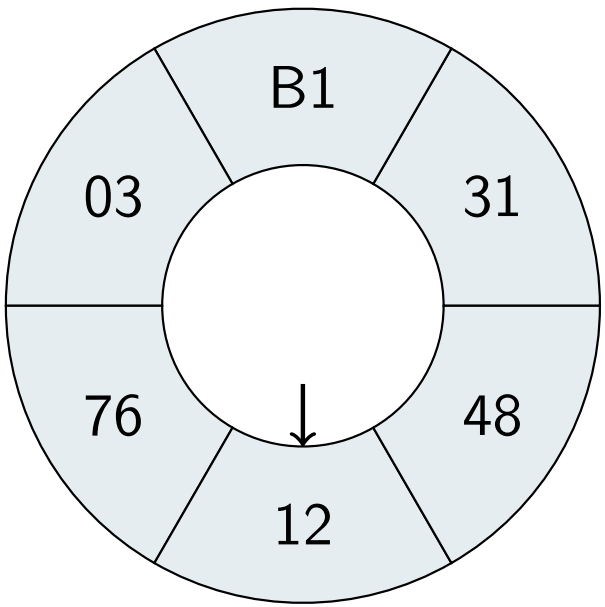
Principe

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



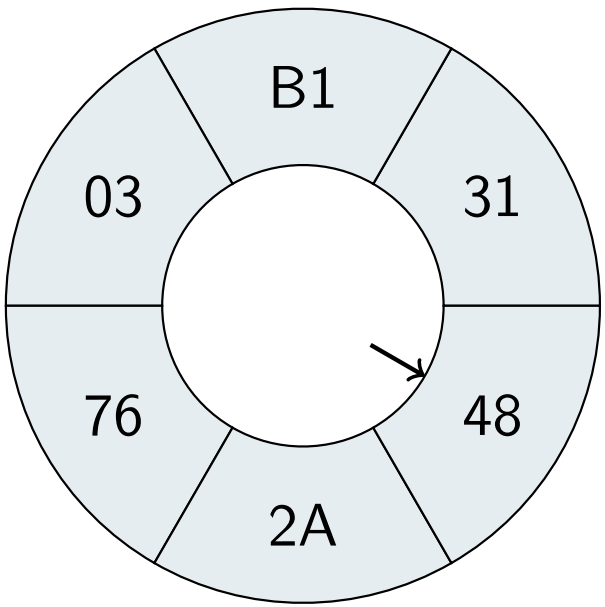
Principe

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



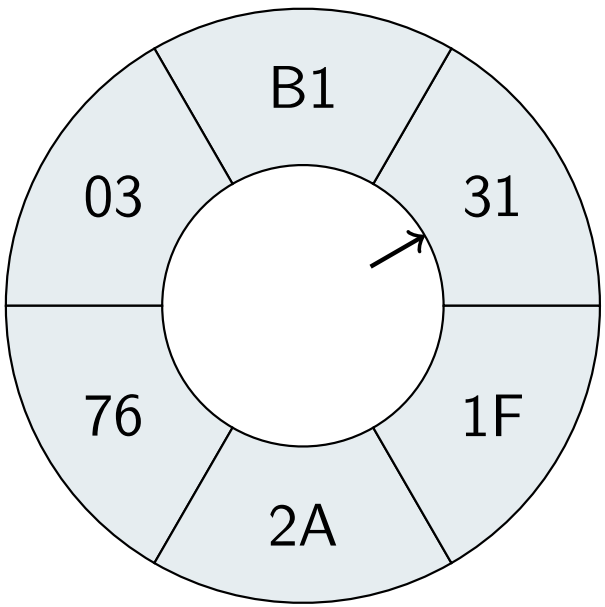
Principe

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



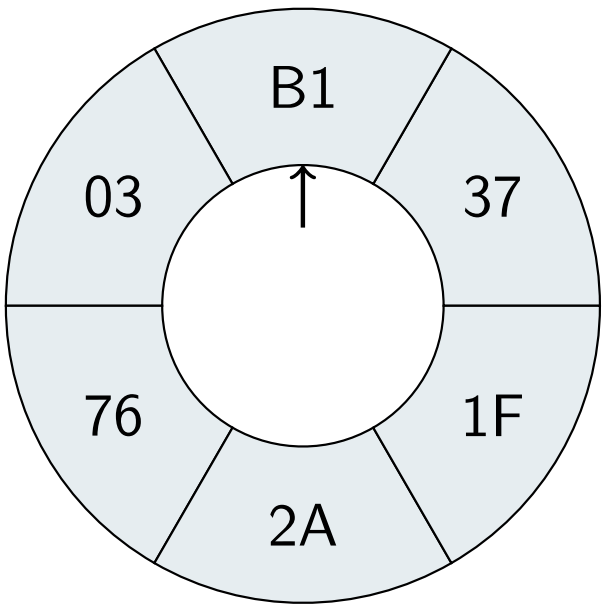
Principe

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



Compter les défauts

	03	04	2A	1F	2A	12	48	31	B1	2A	03	76	2A	1F	37
0	03							31							37
1		04							B1						
2			2A								03				
3				1F								76			
4						12							2A		
5							48							1F	
	*	*	*	*		*	*	*	*		*	*	*	*	*

Total : 13 défauts

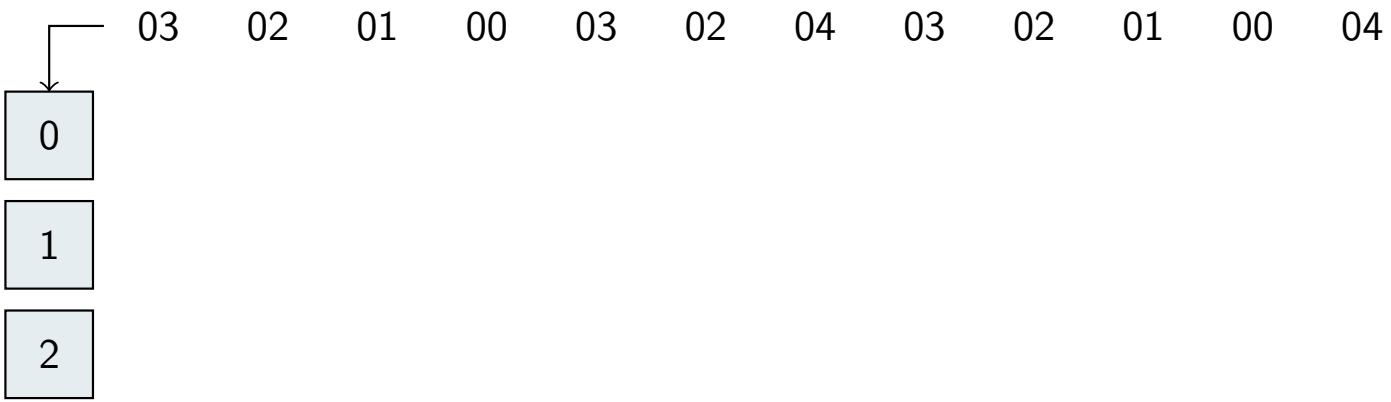
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres $\rightarrow$ plus de défauts !

Exemple : avec 3 cadres



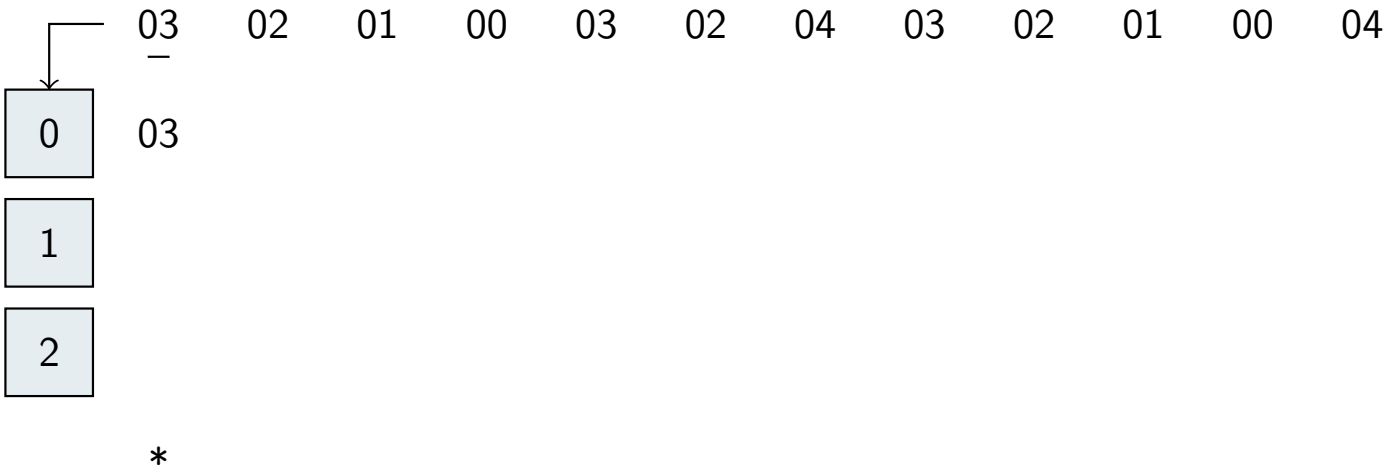
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



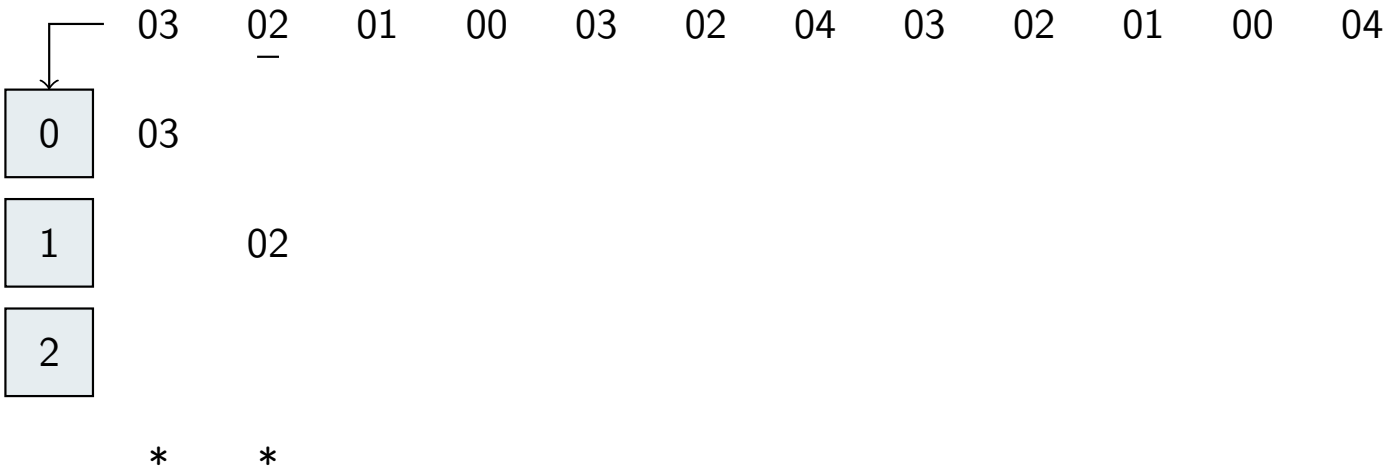
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



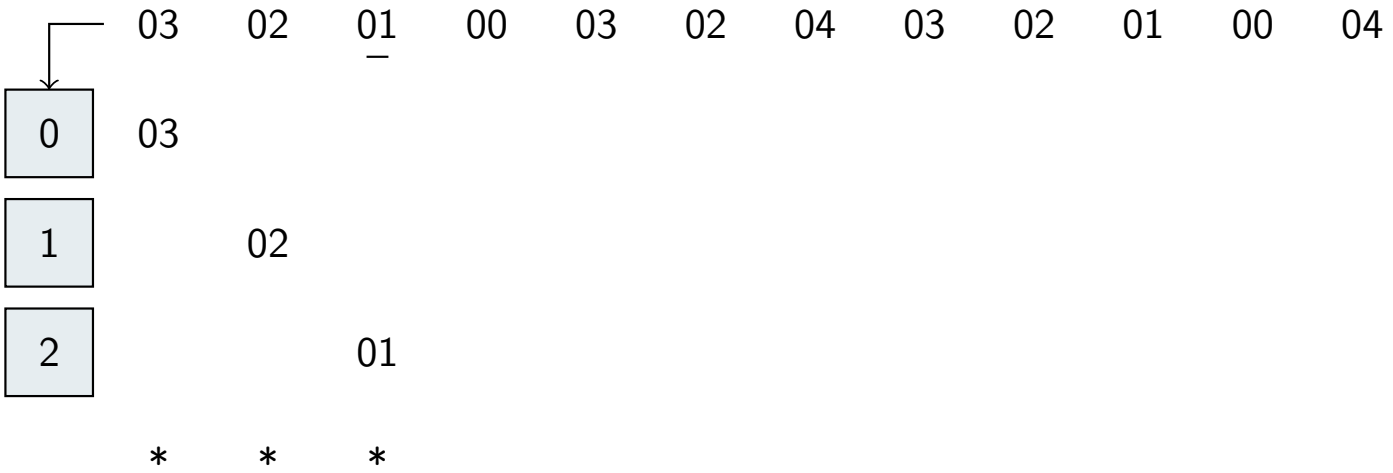
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00								
1		02										
2			01									
	*	*	*	*								

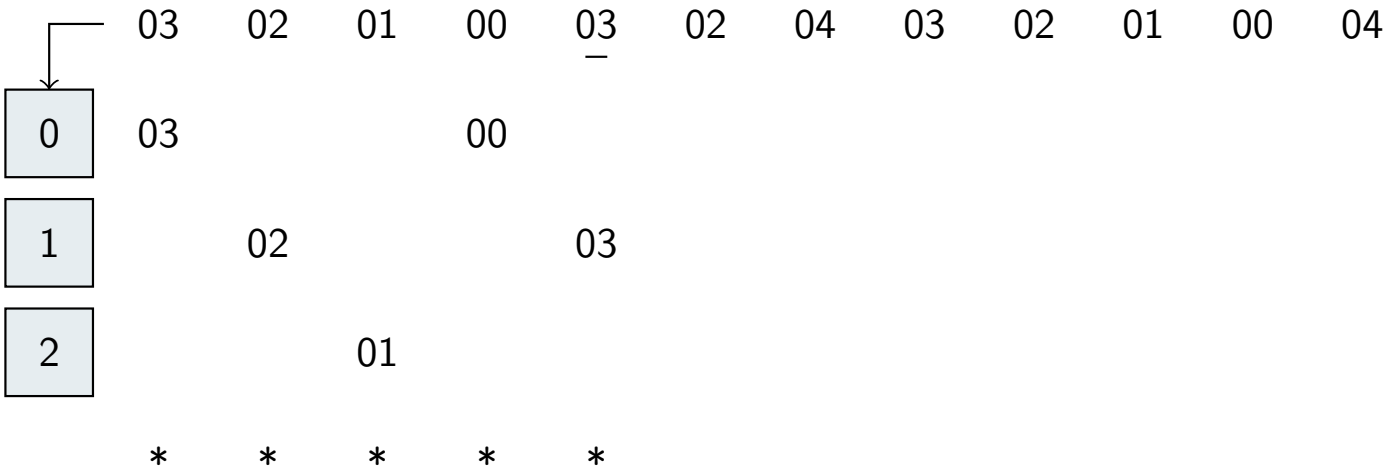
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00								
1		02			03							
2			01			02						
	*	*	*	*	*	*						

Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03							
2			01			02						
	*	*	*	*	*	*	*					

Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03							
2			01			02						
	*	*	*	*	*	*	*					

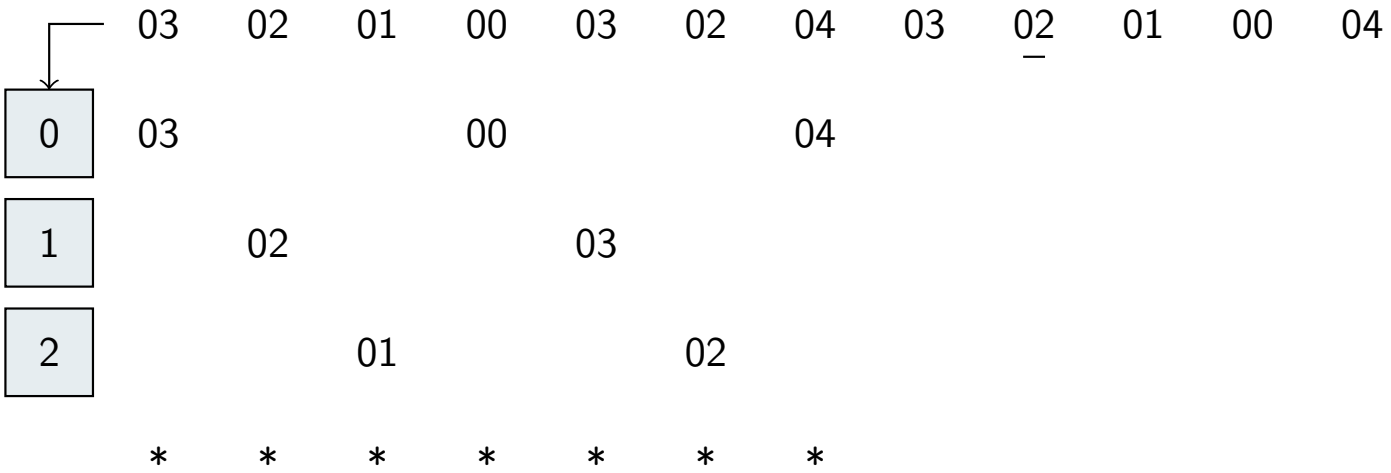
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03					01		
2			01			02						
	*	*	*	*	*	*	*			*		

Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03					01		
2			01			02					00	
	*	*	*	*	*	*	*			*	*	

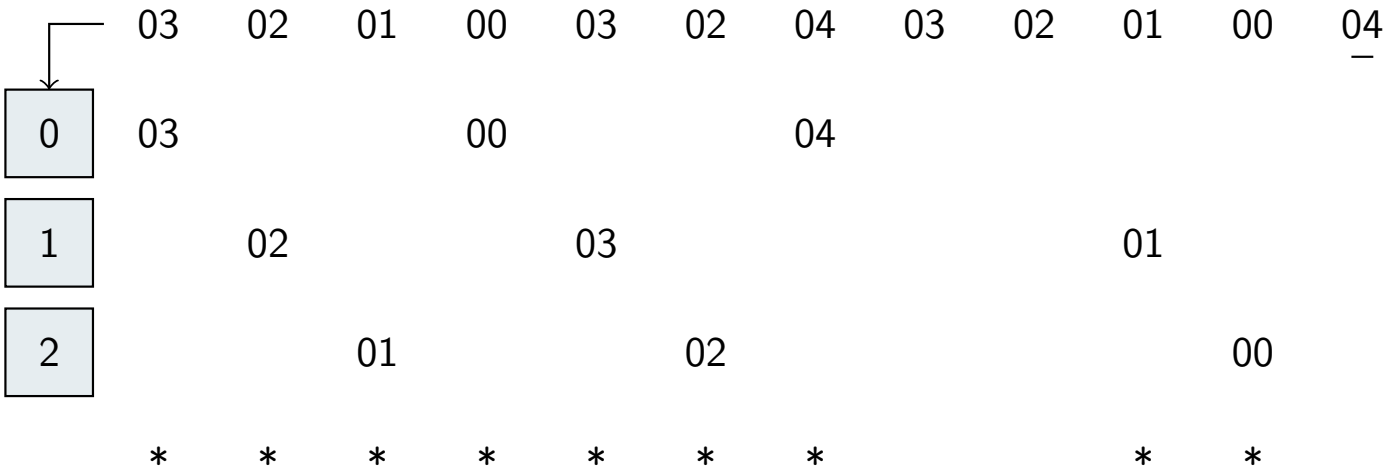
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres



Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03					01		
2			01			02					00	
	*	*	*	*	*	*	*			*	*	

Total : 9 défauts

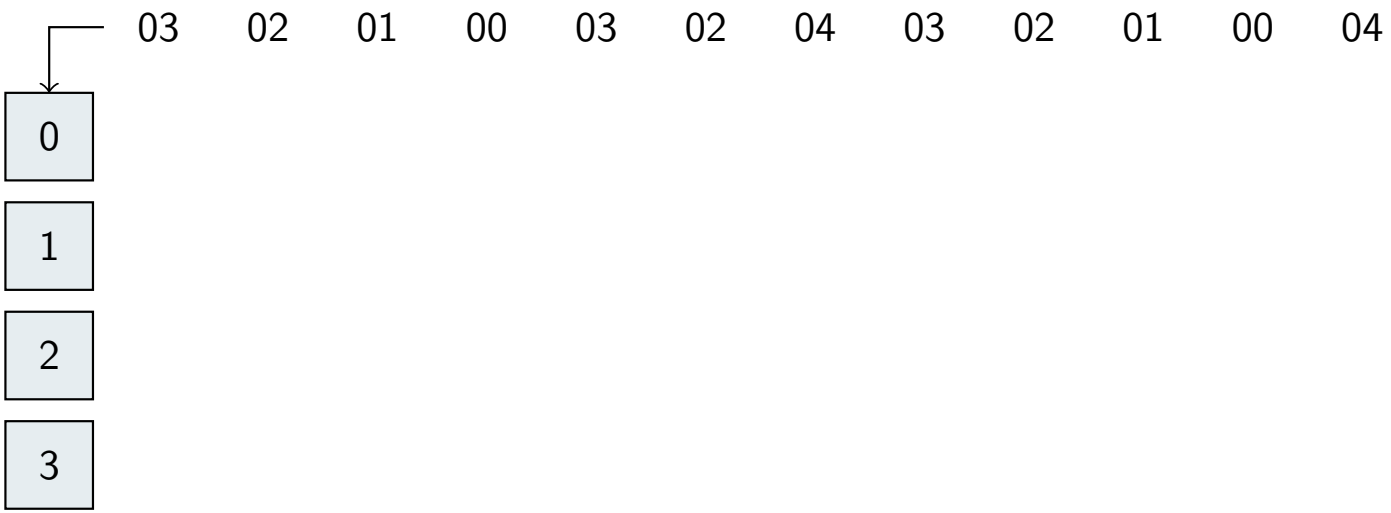
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres $\rightarrow$ plus de défauts !

Exemple : avec 4 cadres



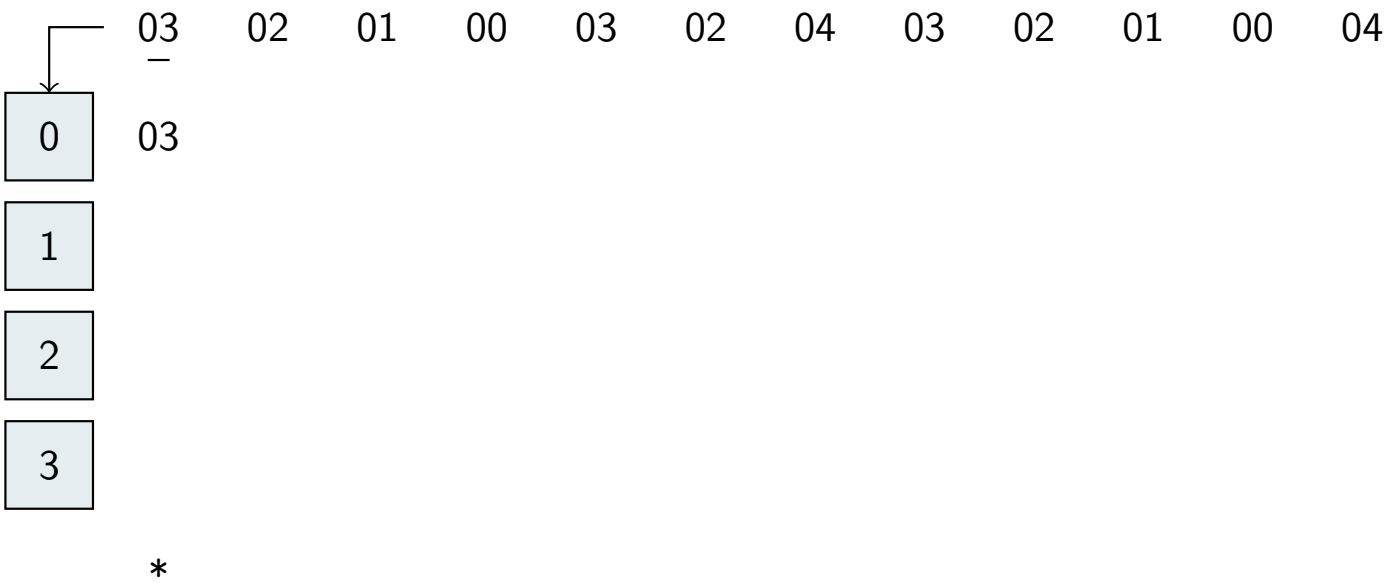
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



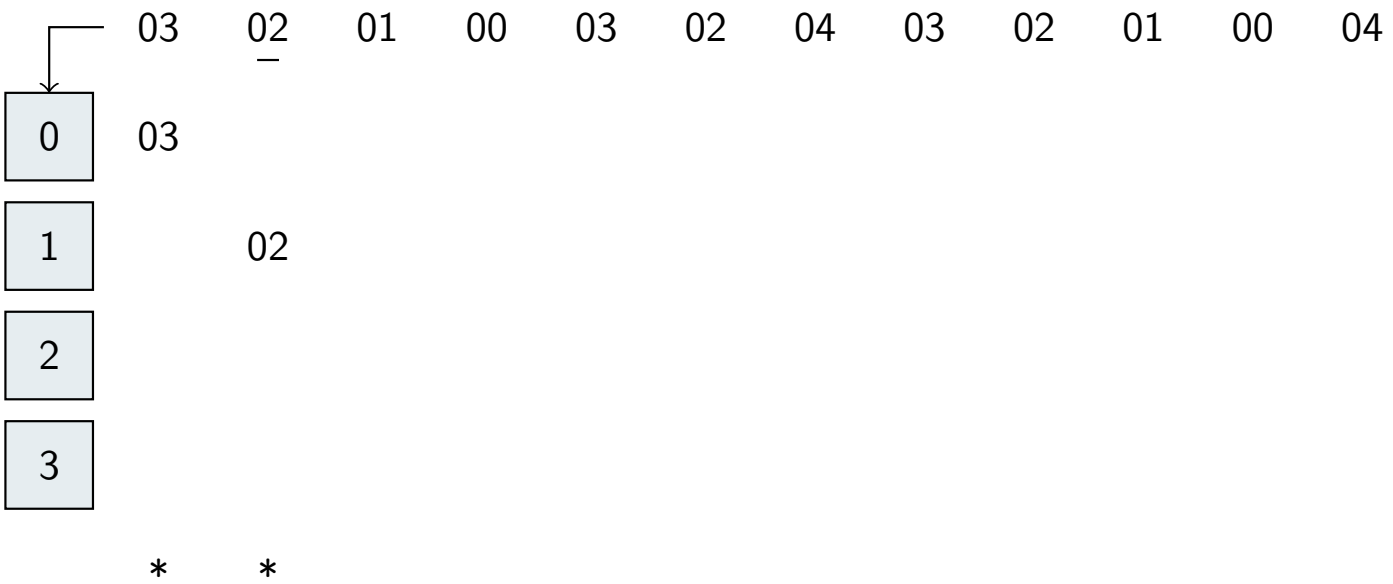
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



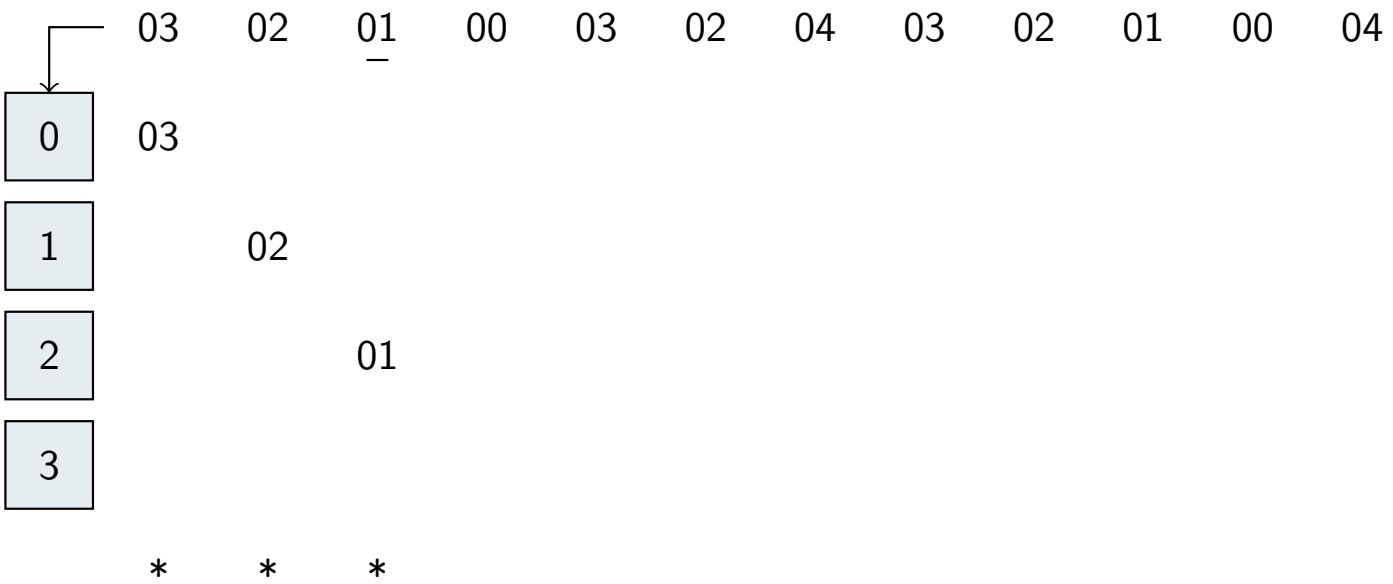
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



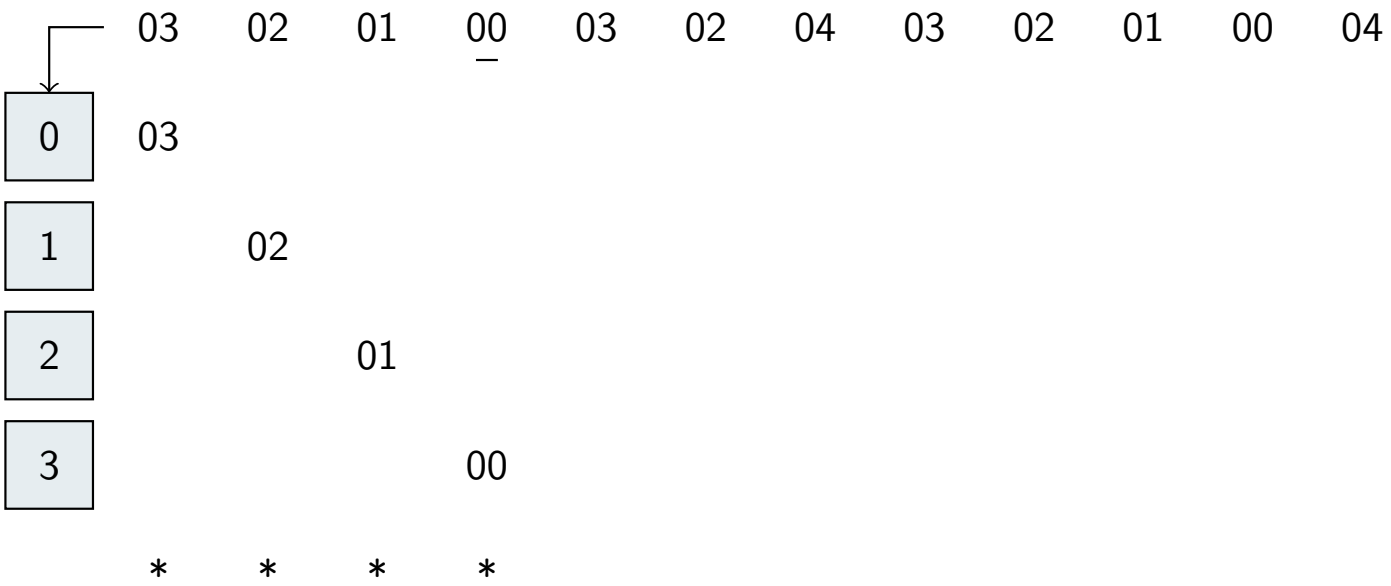
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



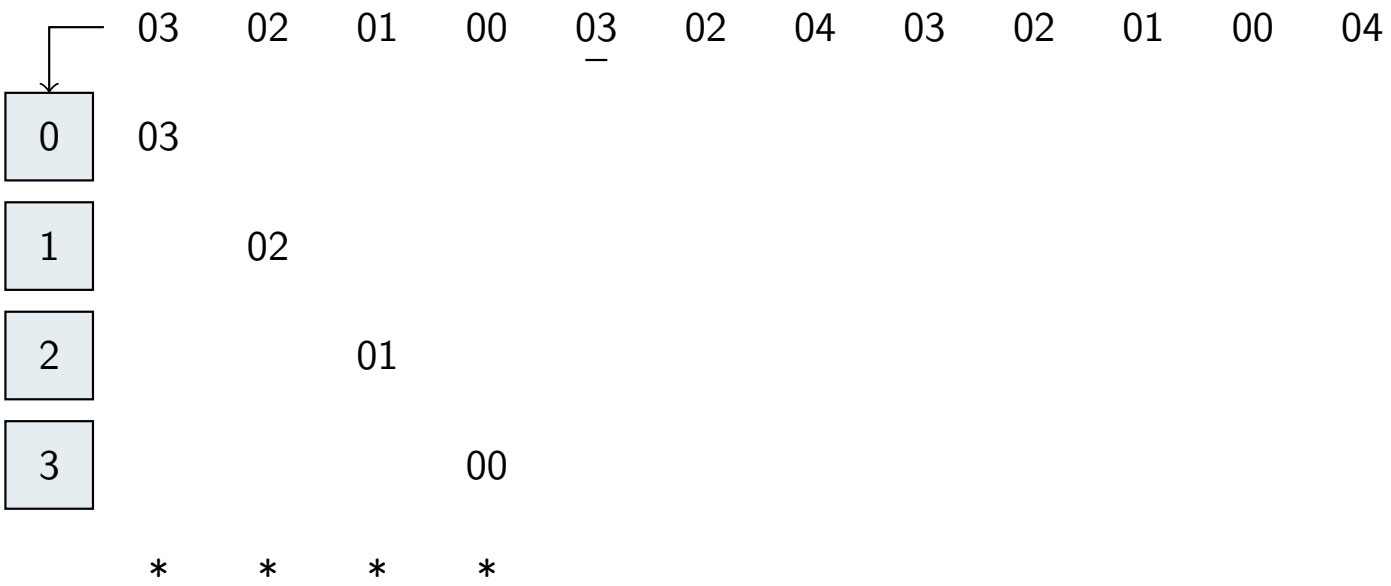
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



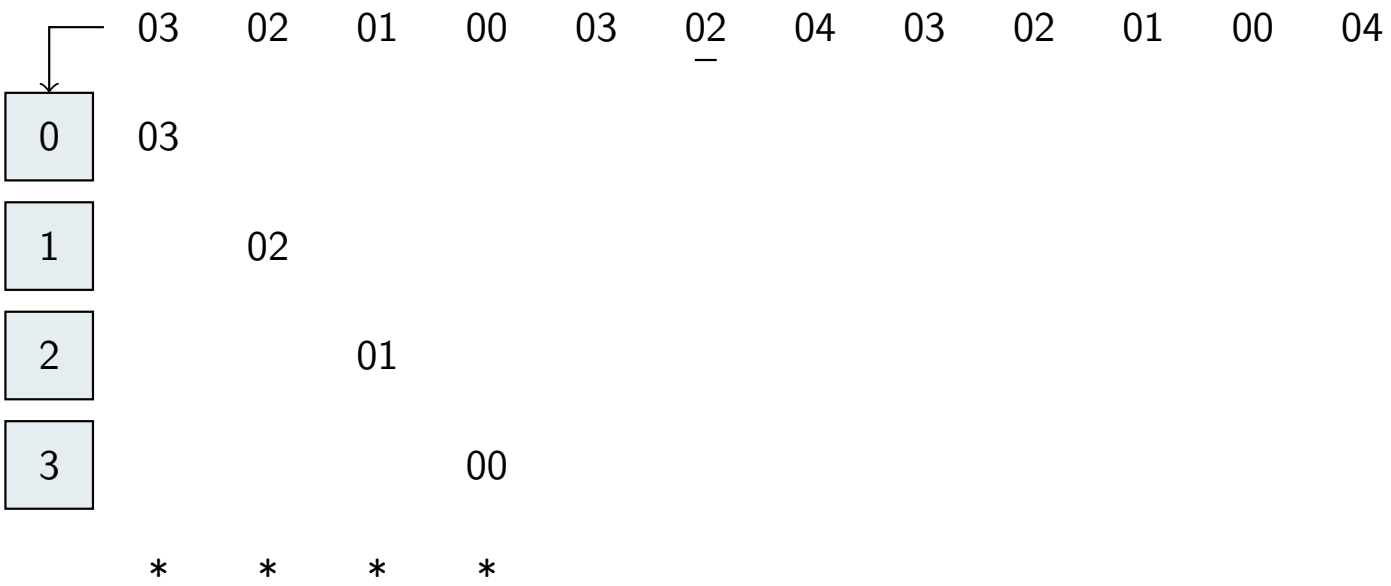
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



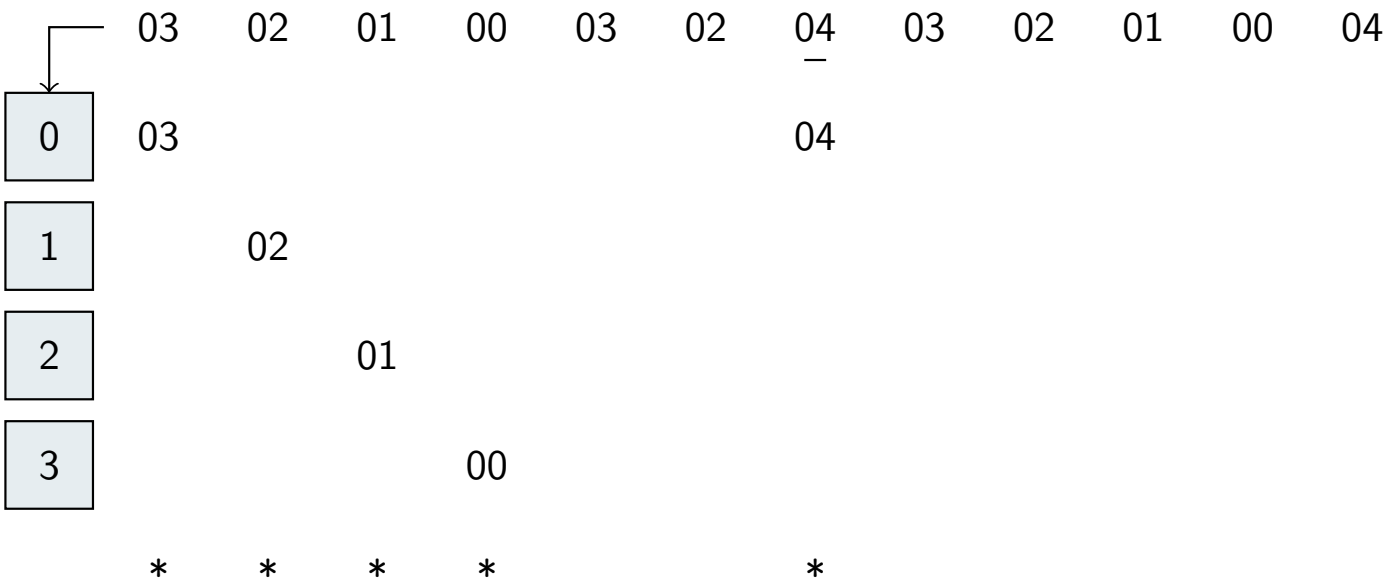
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



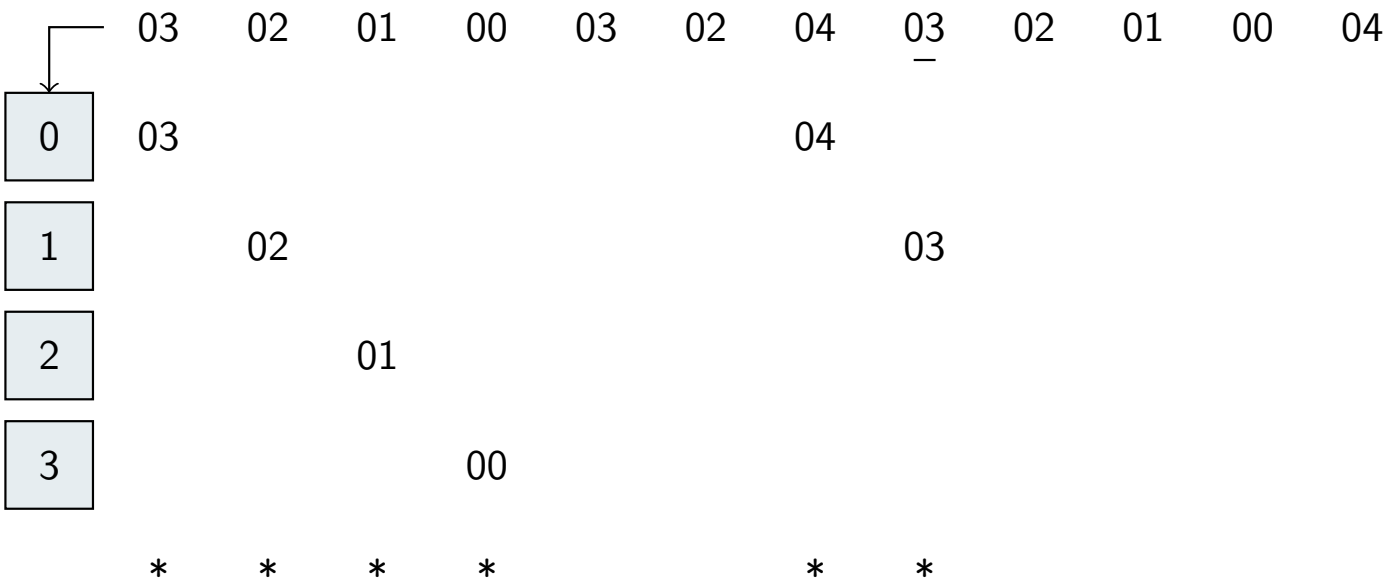
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



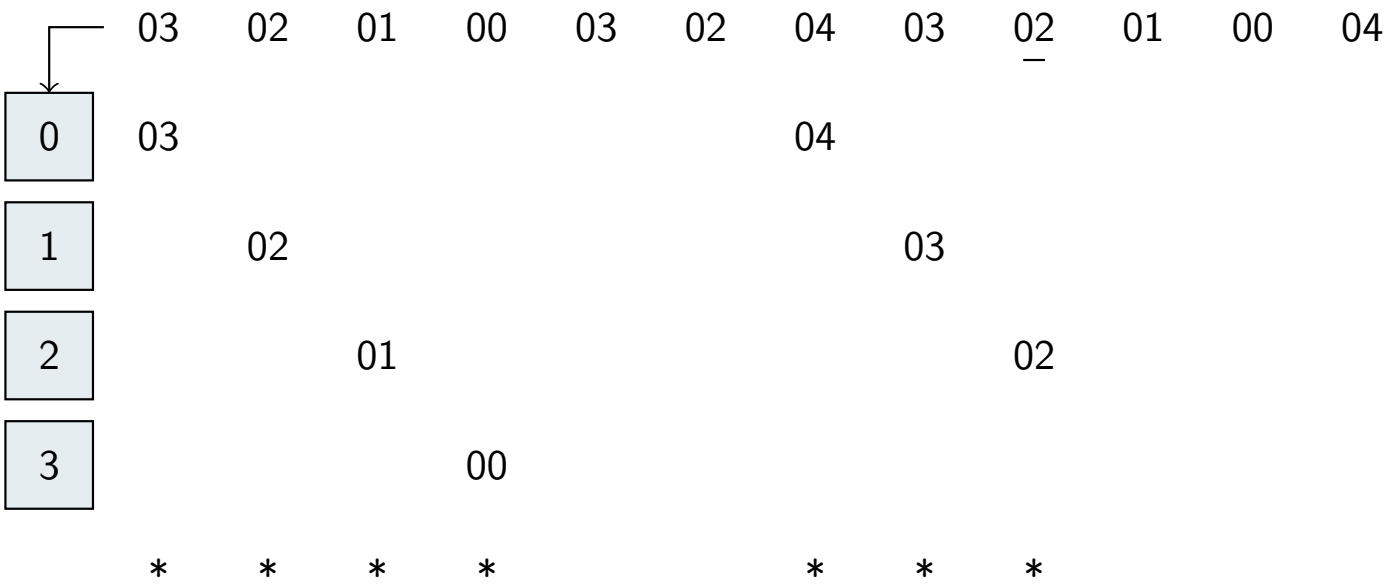
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



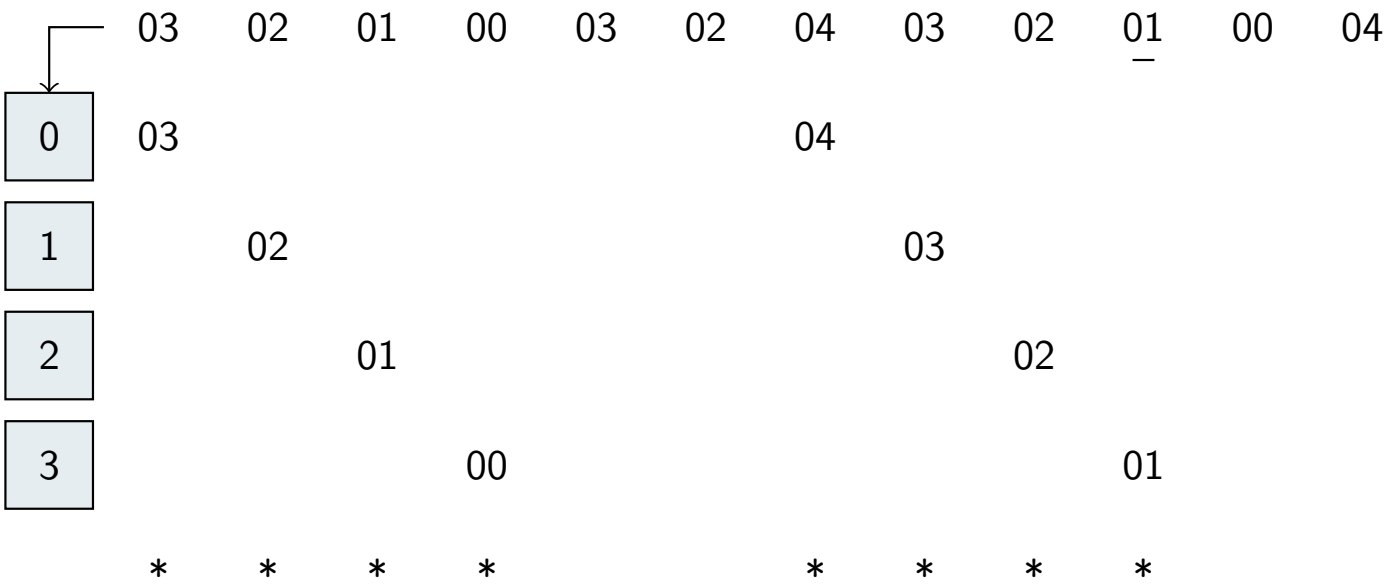
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



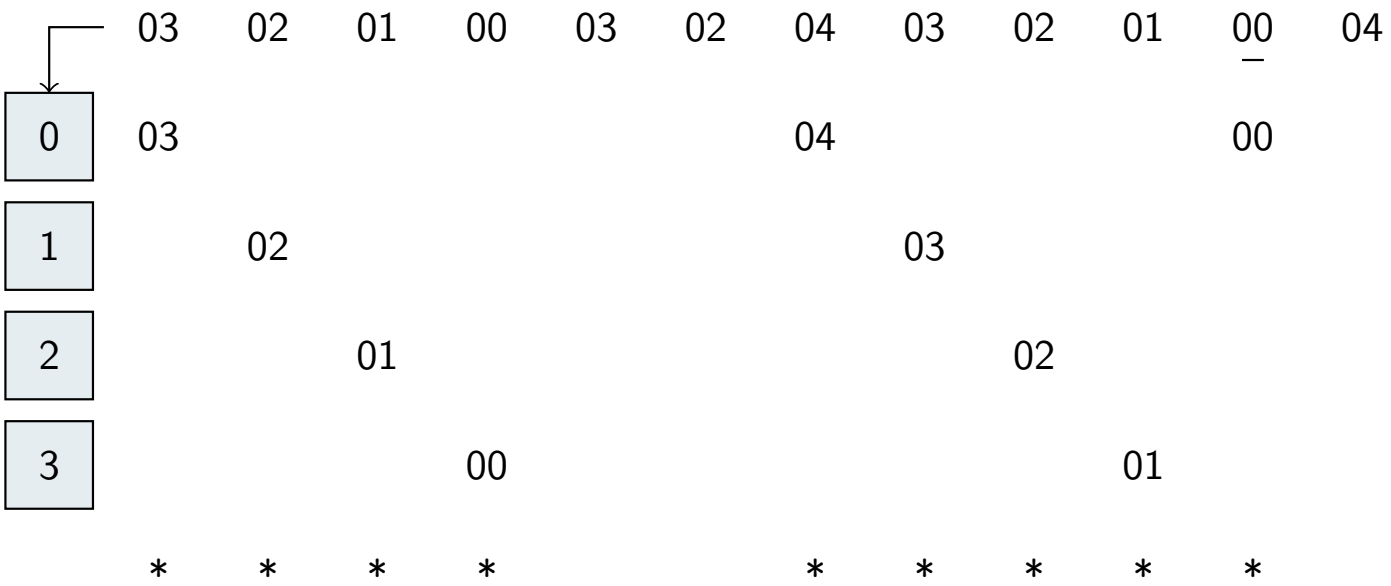
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



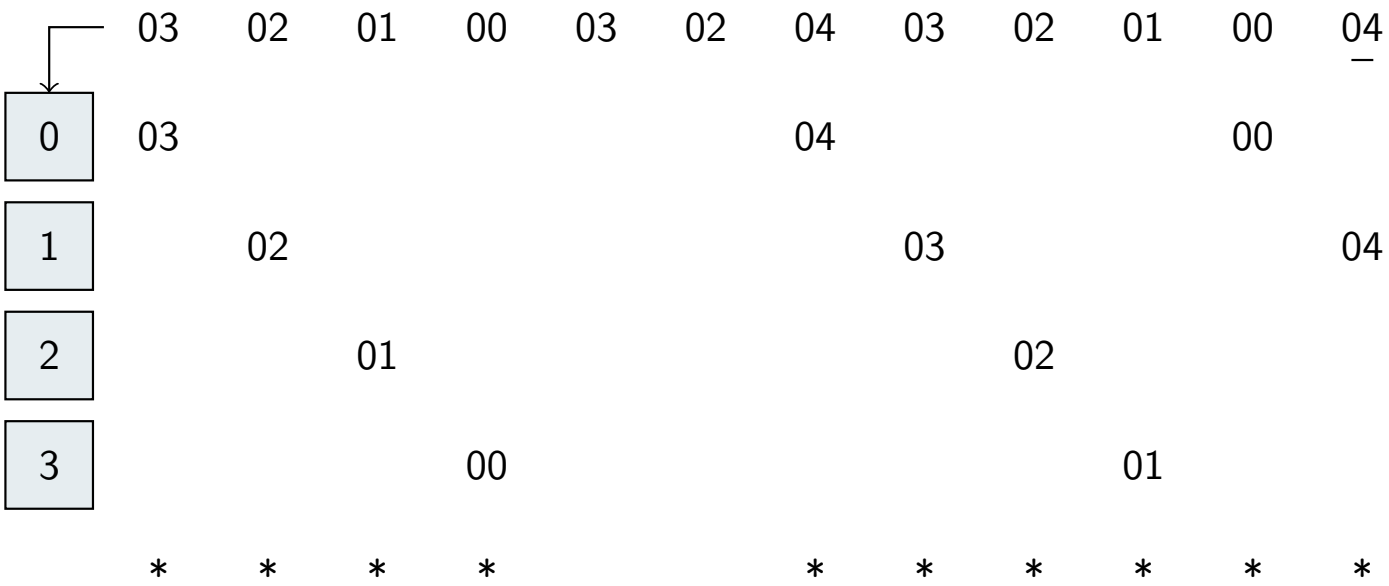
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



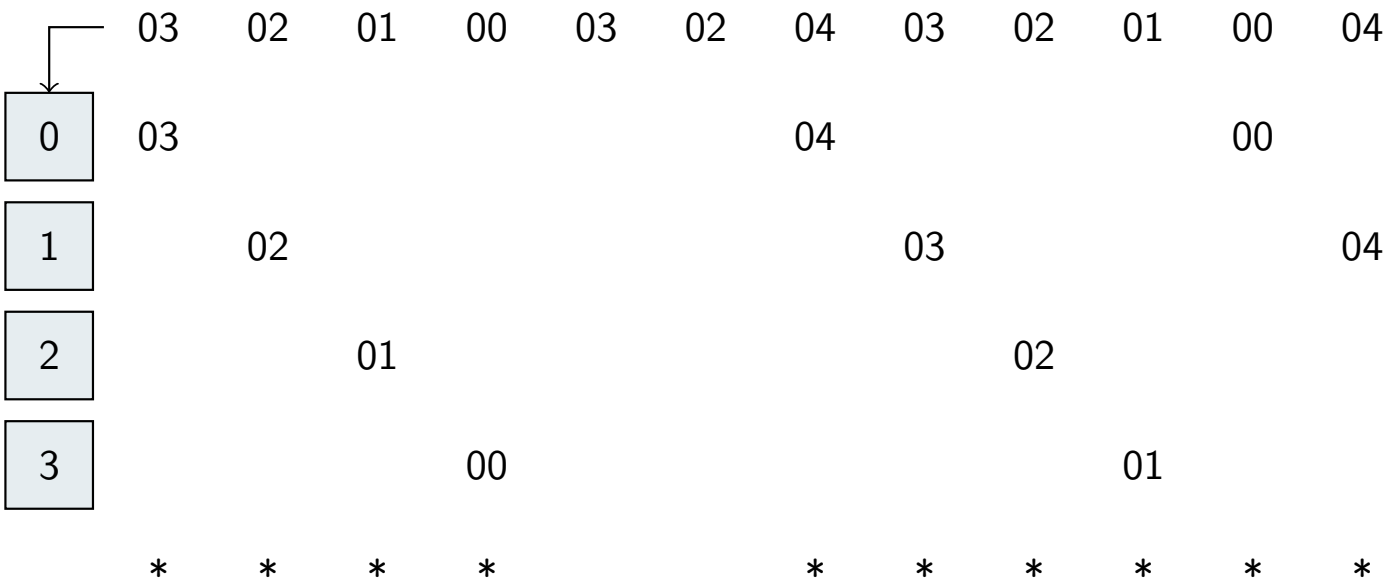
Anomalie de Belady

Algorithmes de type FIFO

sur certaines instances...

Plus de cadres → plus de défauts !

Exemple : avec 4 cadres



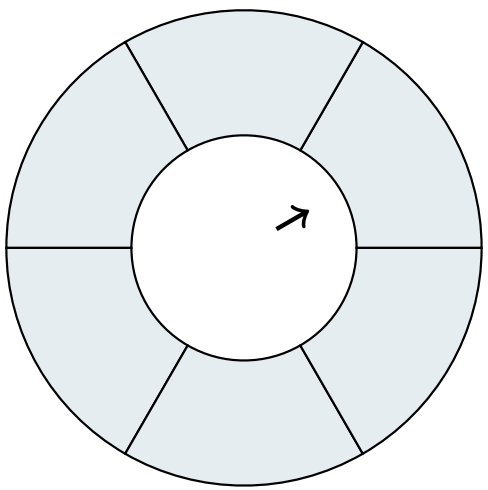
Total : 10 défauts

Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

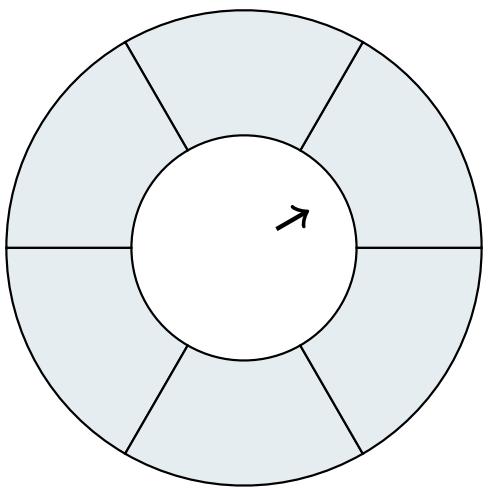


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

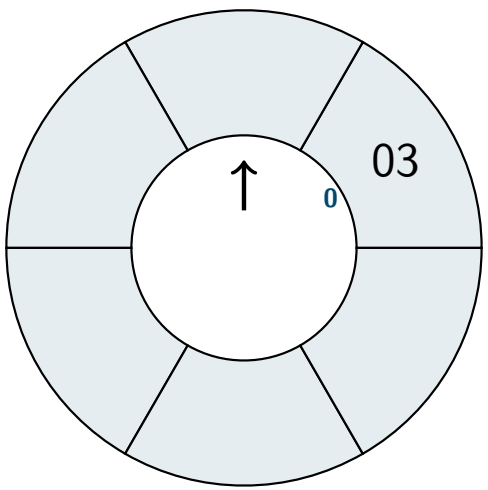


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

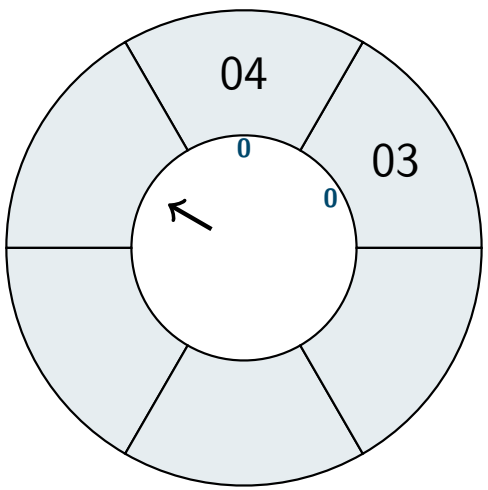


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

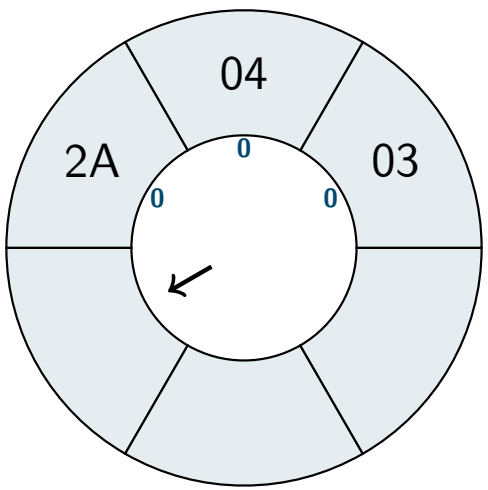


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

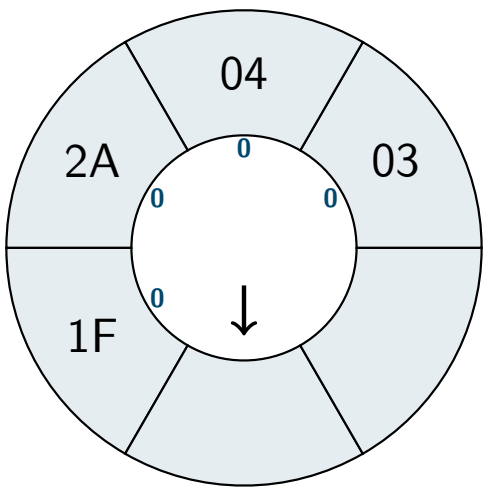


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

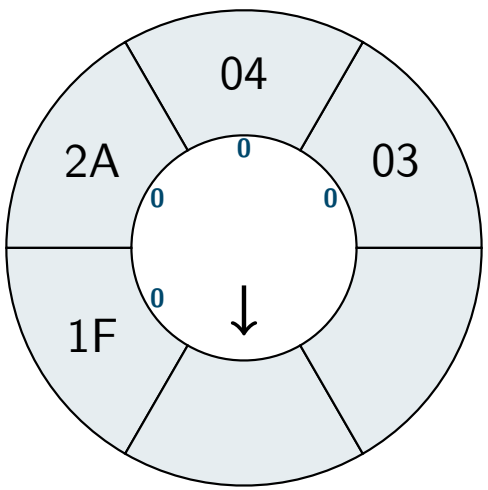


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

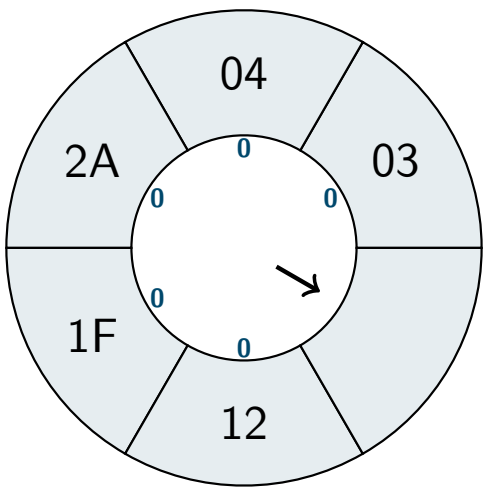


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

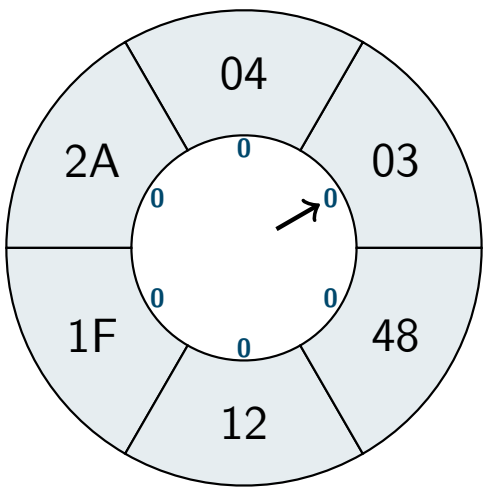


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

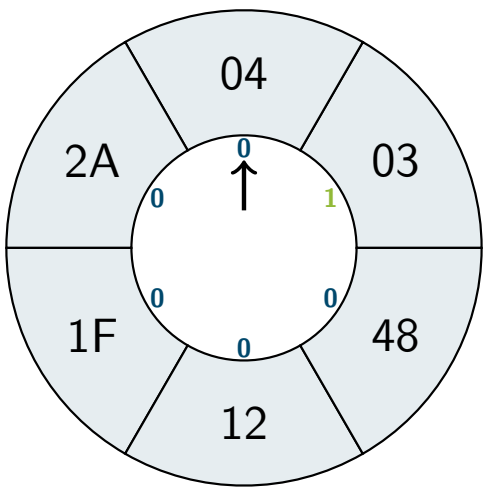


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

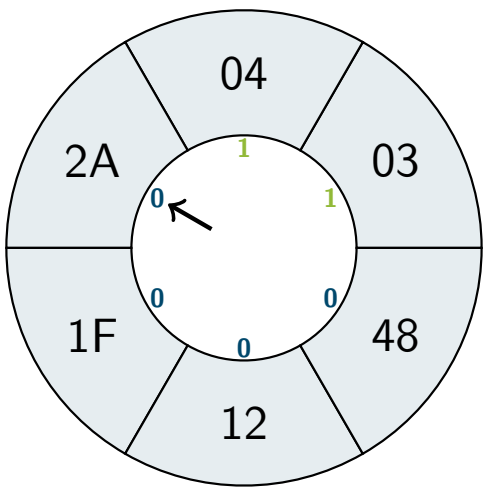


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

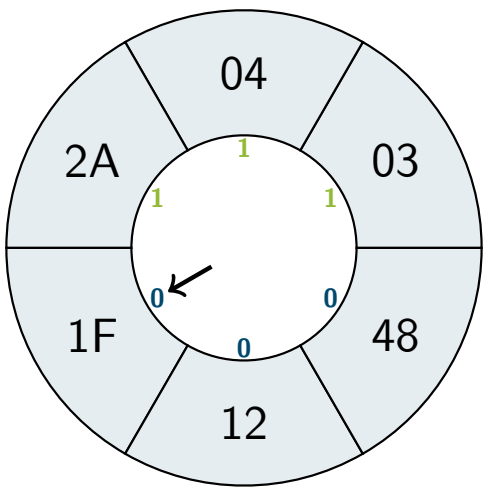


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

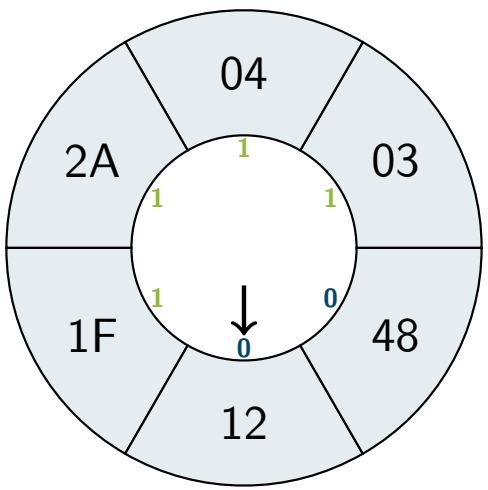


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

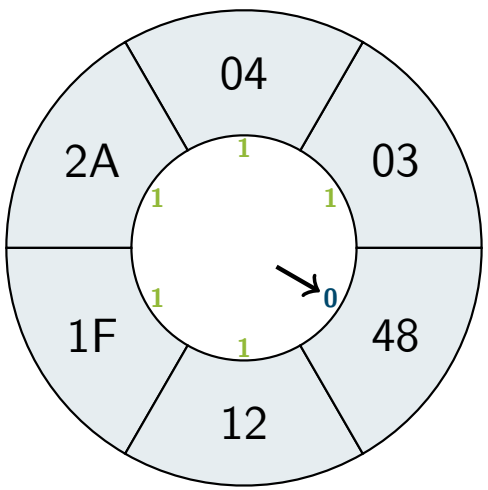


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

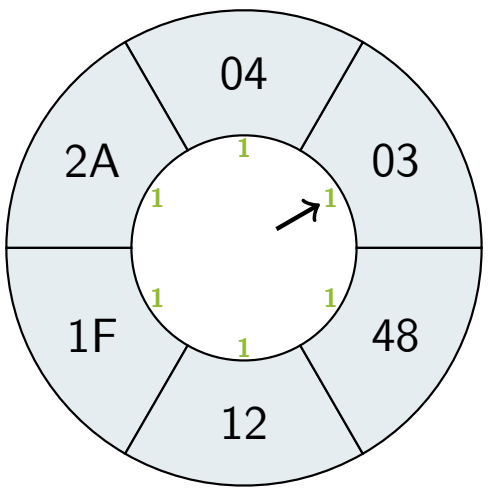


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

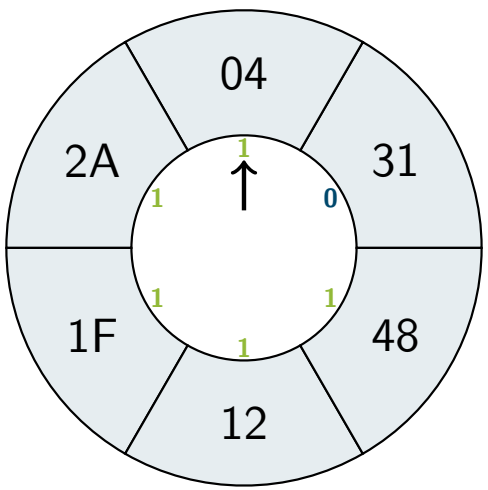


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

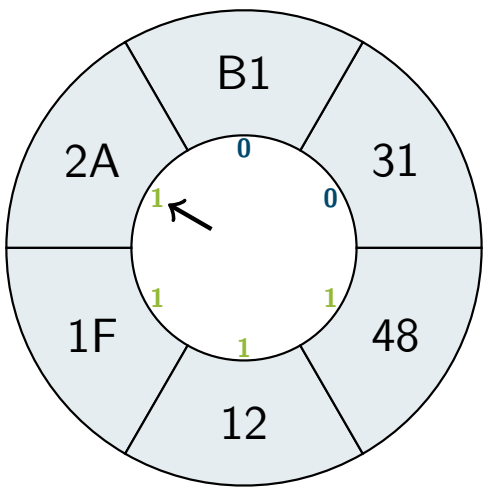


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

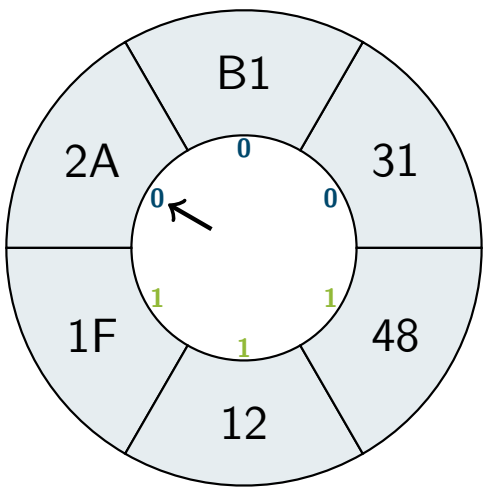


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

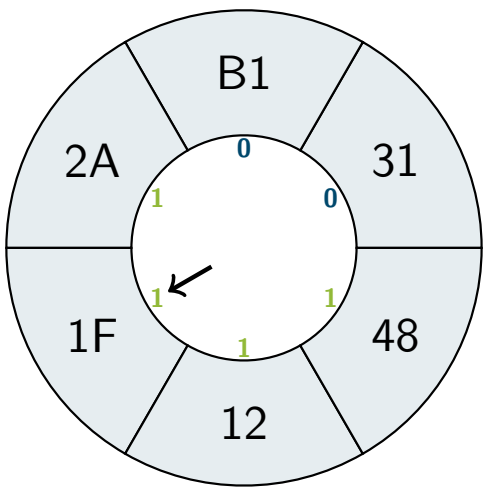


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

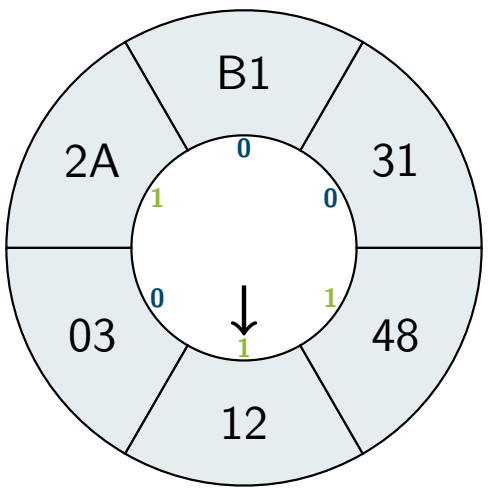


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

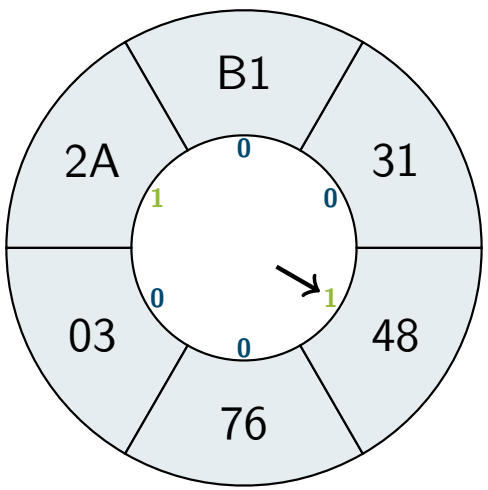


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

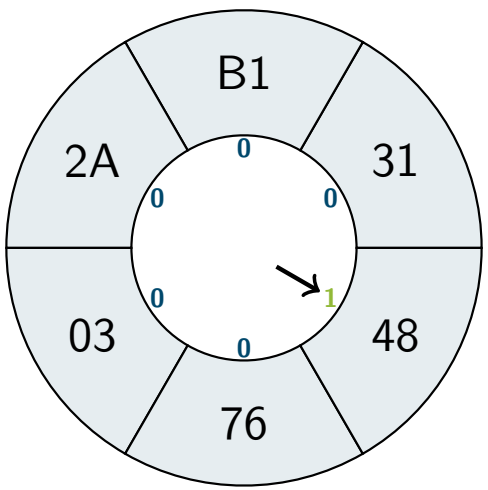


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

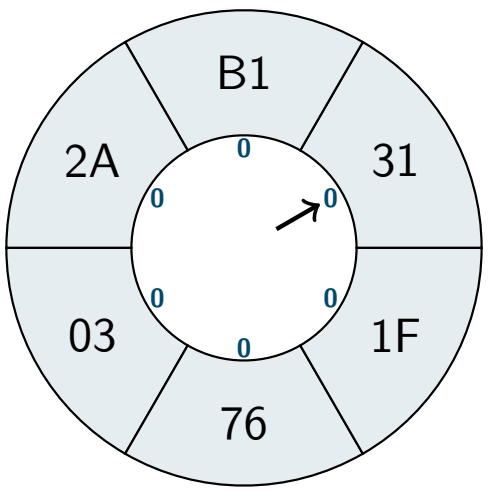


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

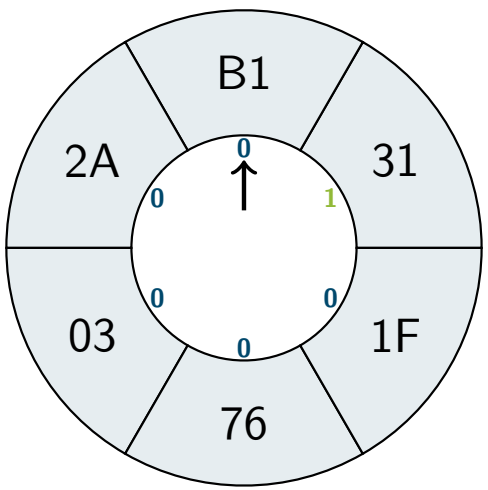


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

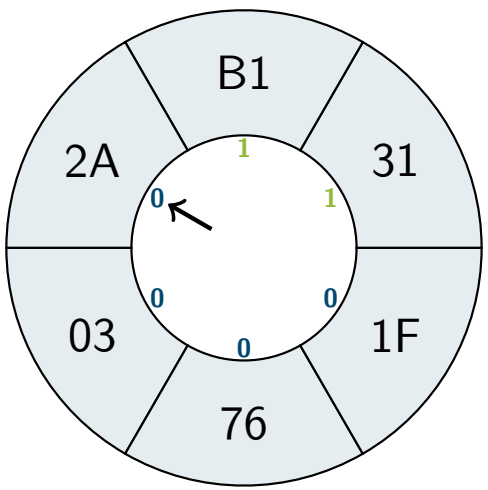


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

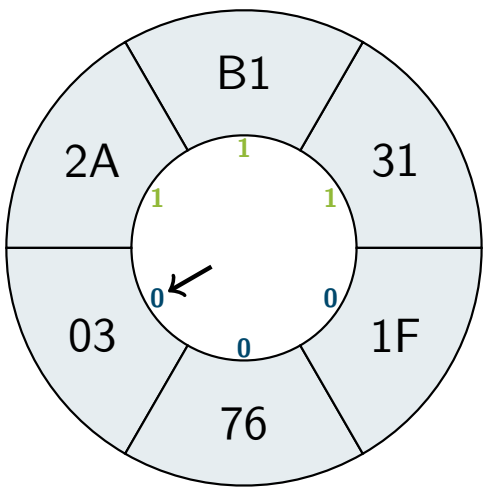


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

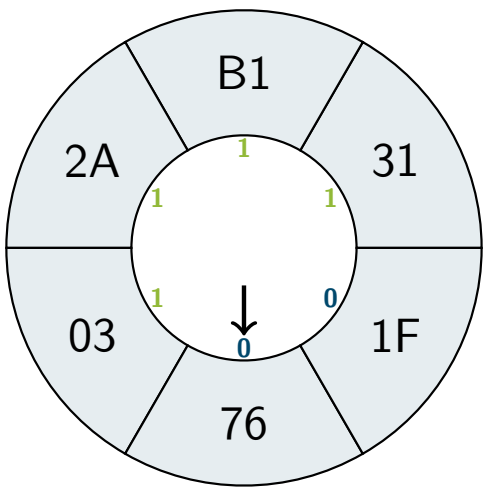


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

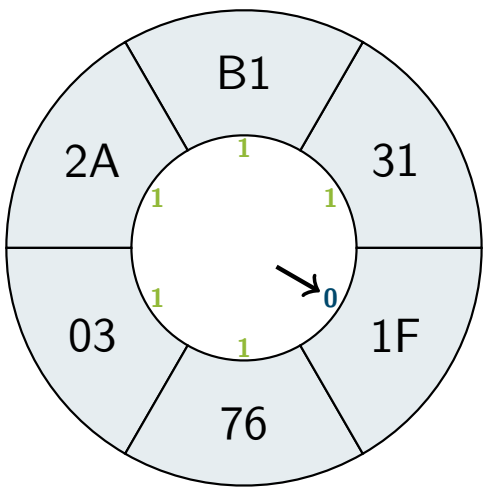


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

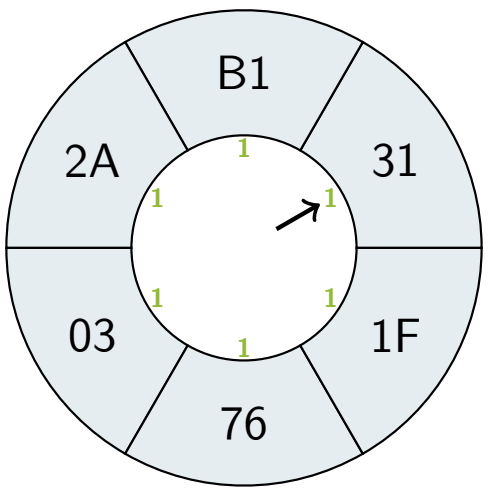


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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

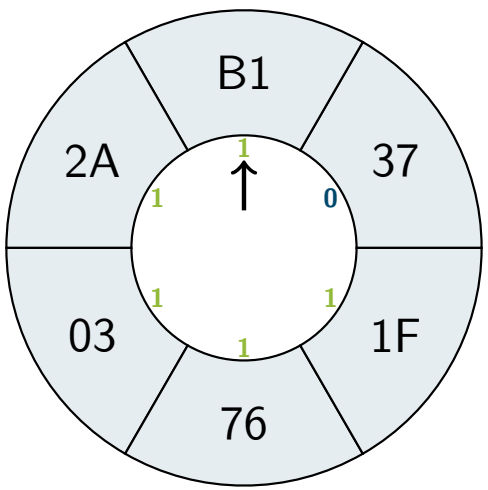


Seconde chance

Principe

- Bit de seconde chance remis à 0 lorsqu'on re-visite une page
- Lorsqu'on a besoin de libérer un cadre :
  - Si bit(cadre_courant)=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



Compter les défauts

	03	04	2A	1F	2A	12	48	31	B1	2A	03	76	2A	1F	37
0	03						+	31						+	37
1		04					+		B1					+	
2			2A		-		+			-	+		-	+	
3				1F			+				03			+	
4						12	+					76		+	
5							48	+						1F	+
	*	*	*	*		*	*	*	*		*	*		*	*

Total : 12 défauts

Principe général

Retirer la page la plus pertinente

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

Principe général

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

Least Frequently Used

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	

Least Frequently Used

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	

Least Frequently Used

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	

Least Frequently Used

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	1	2
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

Least Frequently Used

Principe

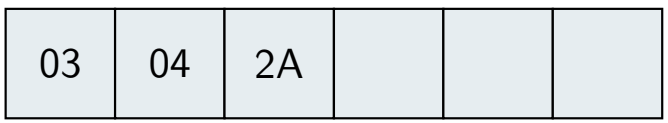
Noter le taux d'utilisation de chaque page du processus.

➜ 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	1	2
12	0	
1F	0	
2A	1	3
31	0	
37	0	
48	0	
76	0	
B1	0	

Least Frequently Used

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	

Least Frequently Used

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	

Least Frequently Used

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	

Least Frequently Used

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	

Least Frequently Used

Principe

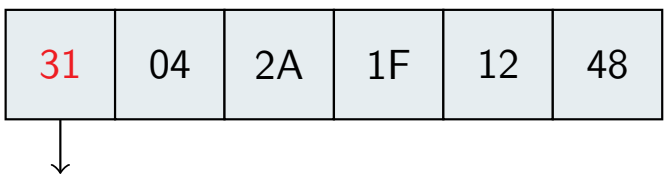
Noter le taux d'utilisation de chaque page du processus.

➜ 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	

Least Frequently Used

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

Least Frequently Used

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

Least Frequently Used

Principe

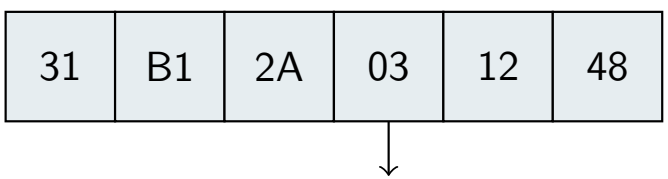
Noter le taux d'utilisation de chaque page du processus.

➜ 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

Least Frequently Used

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

Least Frequently Used

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

Least Frequently Used

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

Least Frequently Used

Principe

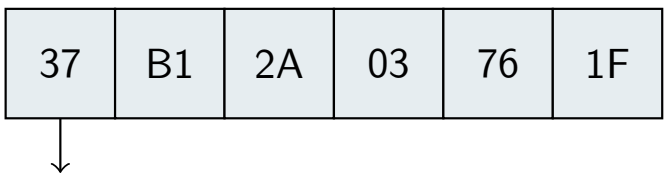
Noter le taux d'utilisation de chaque page du processus.

➜ 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

Compter les défauts

	03	04	2A	1F	2A	12	48	31	B1	2A	03	76	2A	1F	37
0	03							31							37
1		04							B1						
2			2A												
3				1F							03				
4						12						76			
5							48							1F	
	*	*	*	*		*	*	*	*		*	*		*	*

Total : 12 défauts

LFU : avantages et inconvénients

Nombre total d'utilisation

✗ **Problème :** une page beaucoup utilisée reste toujours

LFU : avantages et inconvénients

Nombre total d'utilisation

✗ **Problème :** une page beaucoup utilisée reste toujours

✓ **Solution :** remettre usage à 0 périodiquement

fenêtre glissante trop coûteux à implémenter. . .

LFU : avantages et inconvénients

Nombre total d'utilisation

✗ **Problème :** une page beaucoup utilisée reste toujours

✓ **Solution :** remettre usage à 0 périodiquement

fenêtre glissante trop coûteux à implémenter...

Performance

✓ Beaucoup moins de défaut de page que les autres méthodes

LFU : avantages et inconvénients

Nombre total d'utilisation

✗ **Problème :** une page beaucoup utilisée reste toujours

✓ **Solution :** remettre usage à 0 périodiquement

fenêtre glissante trop coûteux à implémenter...

Performance

✓ Beaucoup moins de défaut de page que les autres méthodes

✗ Temps de calcul + mémoire

Mémoire

✗ 64 bits de plus dans chaque ligne de la table des pages

Temps

✗ $\mathcal{O}(n)$ opérations à chaque page manquante

Not Recently Used

Observations

- Compter les utilisation est coûteux
  $\rightarrow$ *FIFO-2* = 1 seul bit : mémoire + temps !

Not Recently Used

Observations

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

Not Recently Used

Observations

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

Not Recently Used

Observations

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

Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- Tous les R sont remis à 0 chaque K cycles

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

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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	

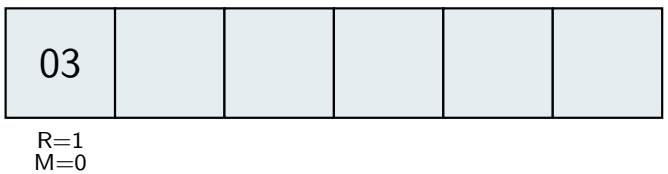
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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	

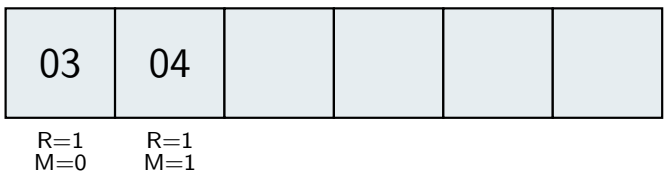
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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	

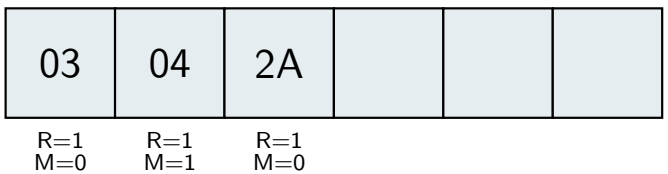
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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	

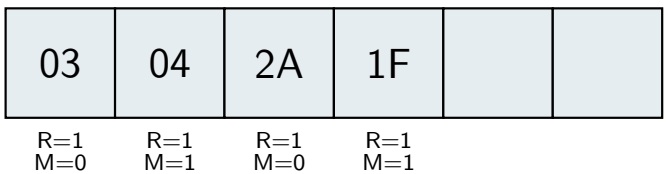
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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	

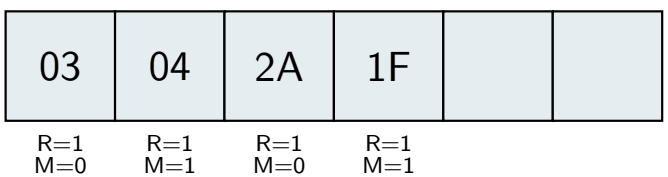
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- Tous les R sont remis à 0 chaque K cycles

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



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	

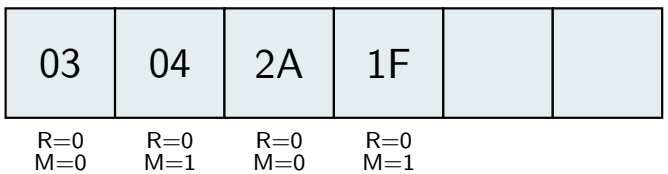
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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



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	

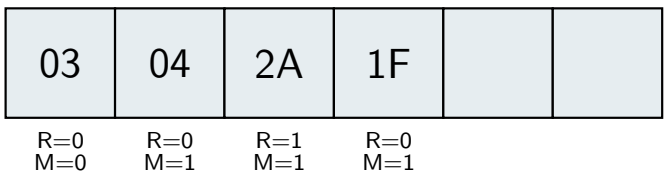
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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	

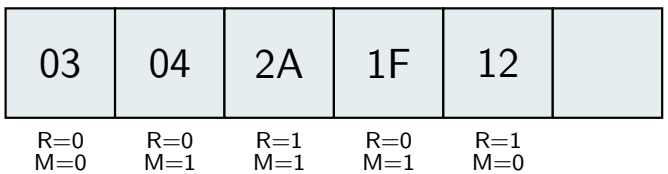
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	

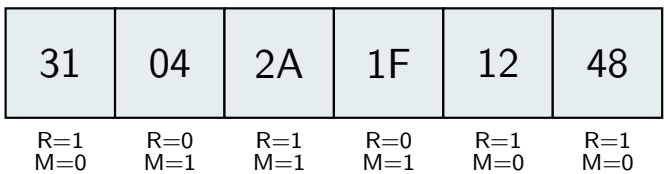
Not Recently Used

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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



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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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	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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- Tous les R sont remis à 0 chaque K cycles

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

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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- Tous les R sont remis à 0 chaque K cycles

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

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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

  FIFO en cas d'égalité
- 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

Principe

- 2 bits : R (page référencée) et M (page modifiée)
- Lecture ou écriture → R=1 ; écriture → M=1
- Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) > (R=0,M=0)

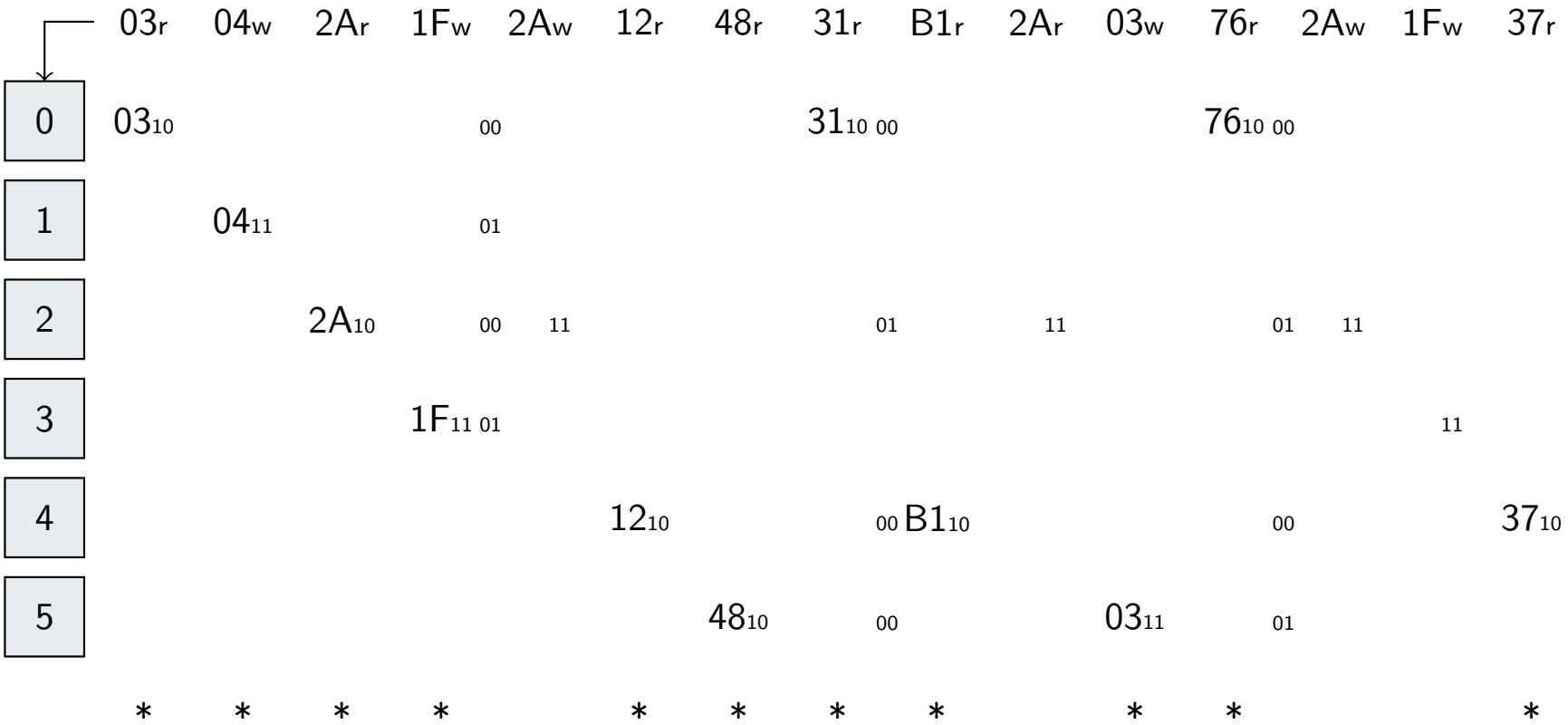
  FIFO en cas d'égalité
- Tous les R sont remis à 0 chaque K cycles

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

Compter les défauts



Total : 11 défauts

NRU : avantages et inconvénients

Performance

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

NRU : avantages et inconvénients

Performance

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

Gain

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

Least Recently Used

Principe

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

Least Recently Used

Principe

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

Implémentation

- Matrice triangulaire de dimension 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						

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

03	04				

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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			

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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			

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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		

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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		

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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		

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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		

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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	

LRU : implémentation

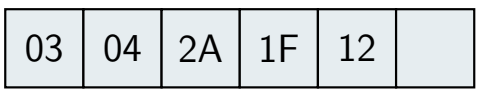
Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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



LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

03	04	2A	1F	12	48

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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	B1	2A	1F	12	48

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

31	B1	2A	03	76	1F

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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
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

LRU : implémentation

Implémentation

- Matrice triangulaire de dimension N = nombre de cadres
- Utilisation d'une page dans le cadre i
  $\rightarrow$ remettre la ligne i à 1 puis la 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

Compter les défauts

	03	04	2A	1F	2A	12	48	31	B1	2A	03	76	2A	1F	37
0	03							31							37
1		04							B1						
2			2A		2A					2A			2A		
3				1F							03				
4						12						76			
5							48							1F	
	*	*	*	*		*	*	*	*		*	*		*	*

Total : 12 défauts

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