



université
PARIS-SACLAY



SYSTÈMES D'EXPLOITATION

ORGANISATION ET GESTION DE LA MÉMOIRE

🎓 3A - Cours Ingénieurs 🏛️ CentraleSupélec 📅 2023/2024



Idir AIT SADOUNE

idir.aitsadoune@centralesupelec.fr

PLAN

- La mémoire
- La mémoire virtuelle
- La mémoire segmentée
- Synthèse

[Retour à l'accueil](#) - [Retour au plan](#)

PLAN

➤ La mémoire

➤ La mémoire virtuelle

➤ La mémoire segmentée

➤ Synthèse

[Retour à l'accueil](#) - [Retour au plan](#)

LA MÉMOIRE POUR QUI ET POURQUOI?

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour le système d'exploitation

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour le système d'exploitation

- Au lancement d'une machine, l'**OS** est le **premier programme** chargé en mémoire.

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour le système d'exploitation

- Au lancement d'une machine, l'**OS** est le **premier programme** chargé en mémoire.
- L'**OS** a besoin d'un **espace mémoire** pour :
 - 👉 le **code** de son **Noyau**
 - 👉 la table des **interruptions**
 - 👉 la table des **processus**
 - 👉 des structures de données (**PCBs** et autres)
 - 👉 ...

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour les processus

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour les processus

- A la création d'un processus, l'**OS crée un PCB** et **alloue de la mémoire** pour le processus.

LA MÉMOIRE POUR QUI ET POURQUOI?

Pour les processus

- A la création d'un processus, l'**OS crée un PCB** et **alloue de la mémoire** pour le processus.
- Pour des raisons de **sécurité**, chaque **processus** doit utiliser **une zone mémoire distincte**.

LA MÉMOIRE POUR QUI ET POURQUOI?

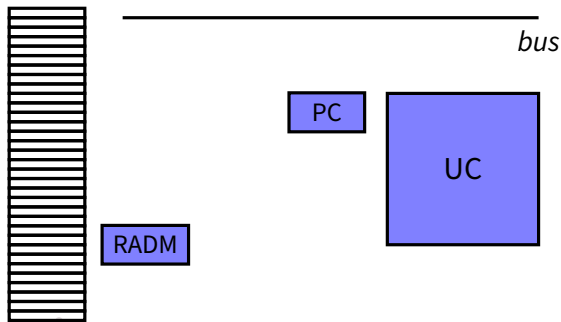
Pour les processus

- A la création d'un processus, l'**OS crée un PCB** et **alloue de la mémoire** pour le processus.
- Pour des raisons de **sécurité**, chaque **processus** doit utiliser **une zone mémoire distincte**.
 - 👉 quel mécanisme d'**allocation** de cet espace?
 - 👉 comment assurer la **protection** de cette zone?
 - 👉 comment assurer la **transparence** de cet espace?

ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

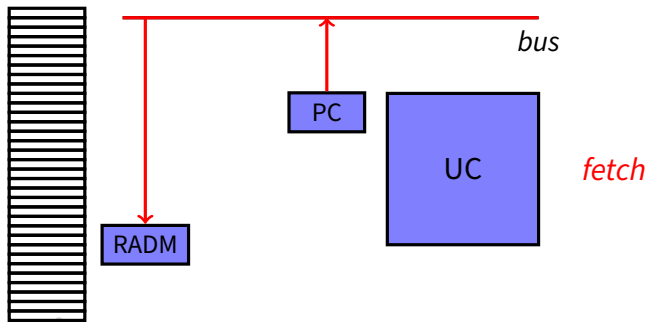
- Des instructions → registre PC sur le processeur
- Des données → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

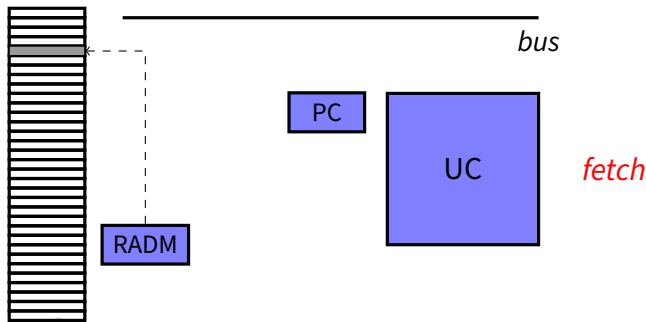
- Des **instructions** → registre PC sur le processeur
- Des données → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

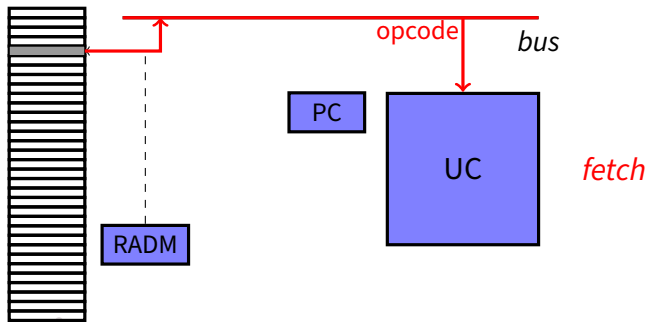
- Des **instructions** → registre PC sur le processeur
- Des données → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

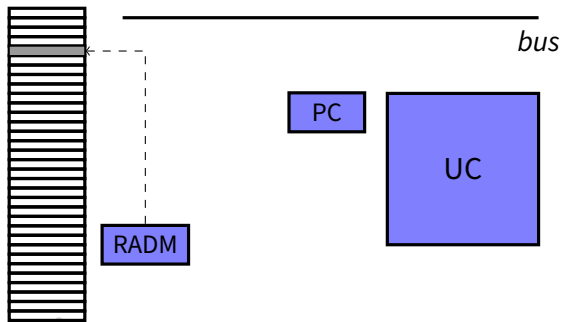
- Des **instructions** → registre PC sur le processeur
- Des données → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

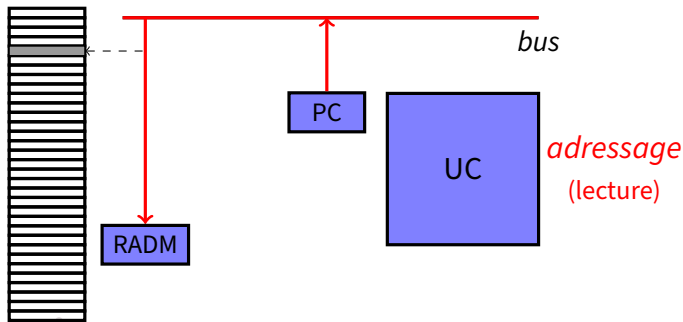
- Des instructions → registre PC sur le processeur
- Des données → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

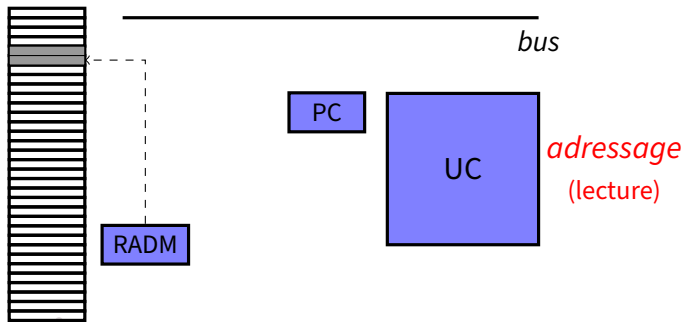
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

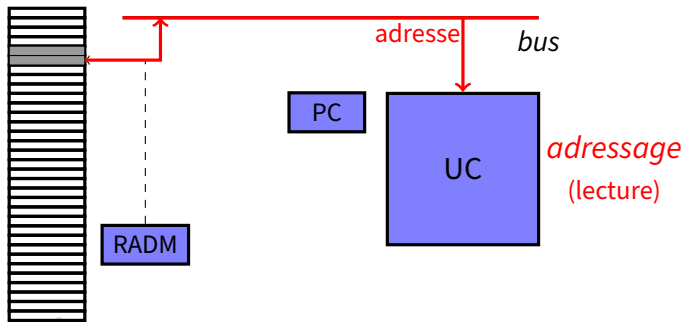
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

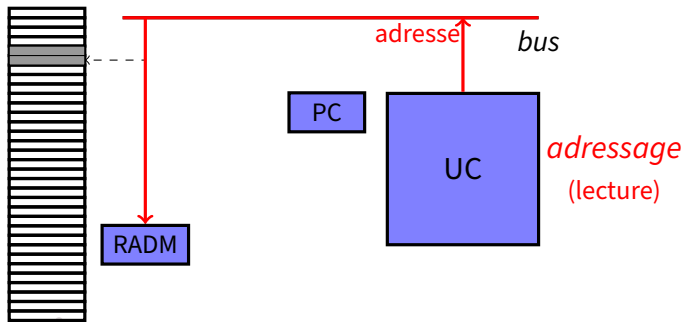
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

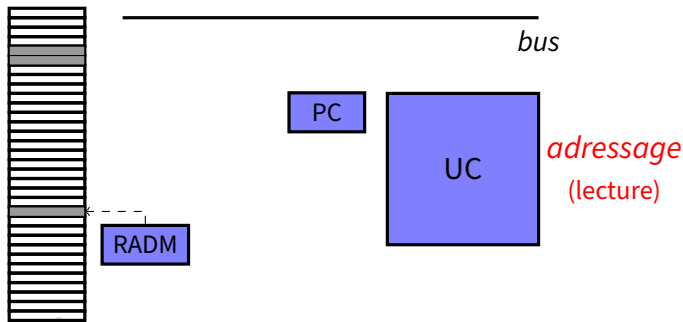
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

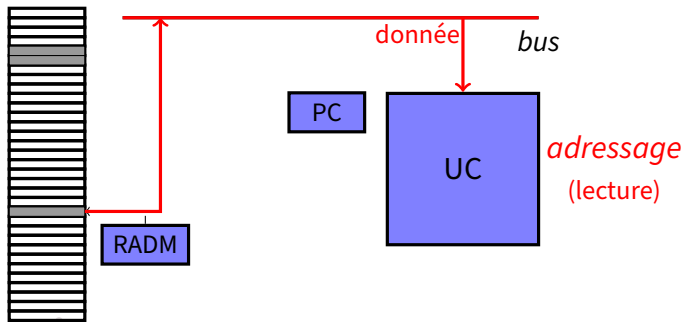
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

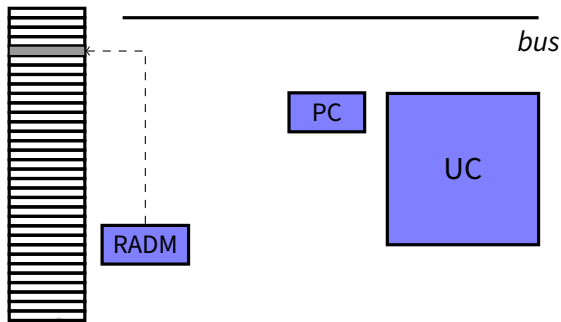
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

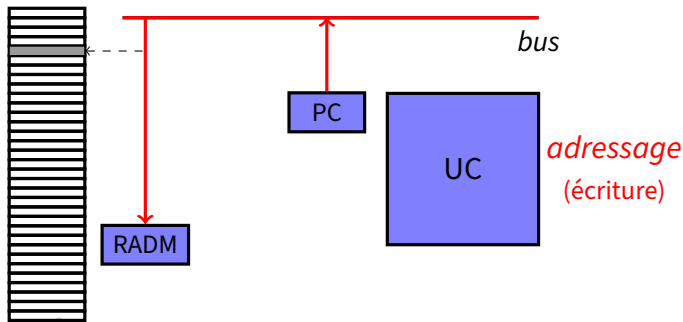
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

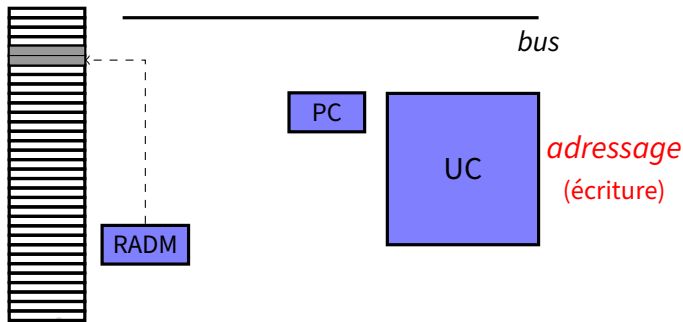
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

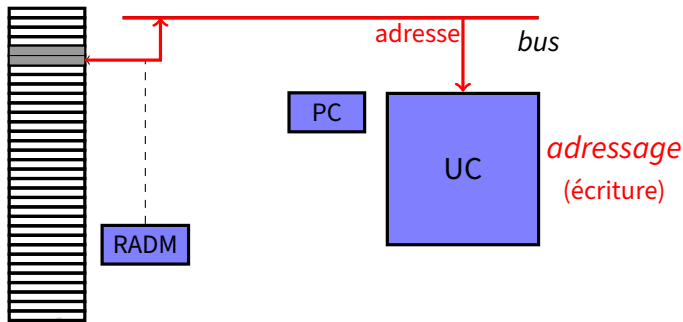
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

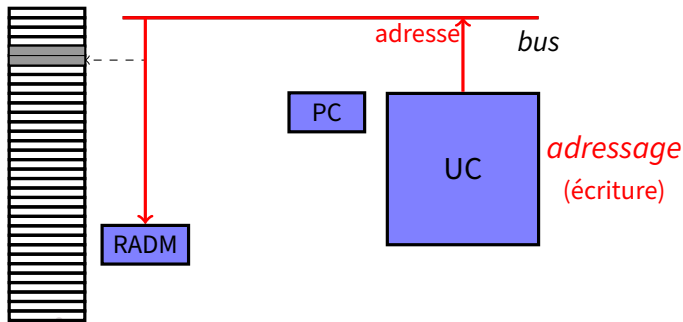
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

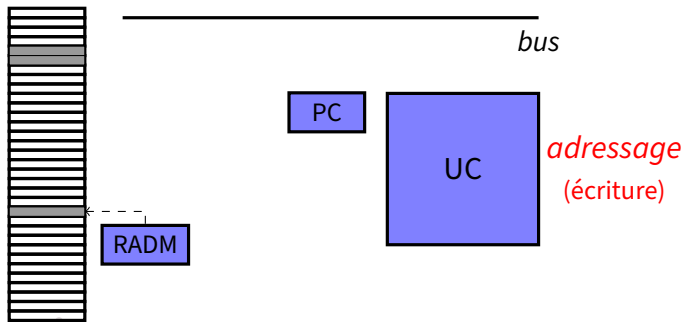
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

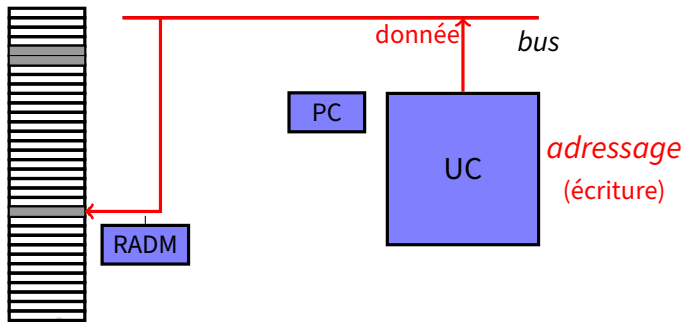
- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** et contenant :

- Des instructions → registre PC sur le processeur
- Des **données** → registre RADM sur le processeur



STRUCTURE DE LA MÉMOIRE

STRUCTURE DE LA MÉMOIRE

- Chaque **case mémoire** est associée à une **adresse**
👉 le numéro de la case

STRUCTURE DE LA MÉMOIRE

- Chaque **case mémoire** est associée à une **adresse**
👉 le numéro de la case
- Cette **adresse** est obtenue depuis **une instruction**
👉 l'adresse est en **binaire** sur n bits

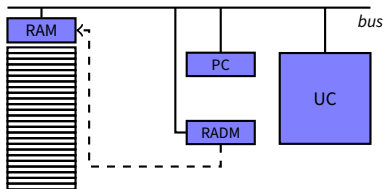
STRUCTURE DE LA MÉMOIRE

- Chaque **case mémoire** est associée à une **adresse**
👉 le numéro de la case
- Cette **adresse** est obtenue depuis **une instruction**
👉 l'adresse est en **binaire** sur n bits
- Il y a donc 2^n **différentes adresses** possibles
👉 2^n cases de $00 \dots 0$ à $11 \dots 1$

STRUCTURE DE LA MÉMOIRE

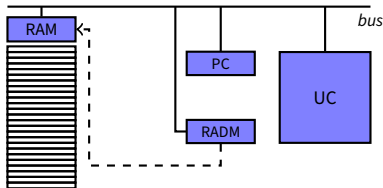
- Chaque **case mémoire** est associée à une **adresse**
👉 le numéro de la case
- Cette **adresse** est obtenue depuis **une instruction**
👉 l'adresse est en **binaire** sur n bits
- Il y a donc 2^n **différentes adresses** possibles
👉 2^n cases de 00...0 à 11...1
- **Exemple** : 32 bits $\rightarrow 2^{32} \approx 4$ Go

FONCTIONNEMENT DE LA MÉMOIRE



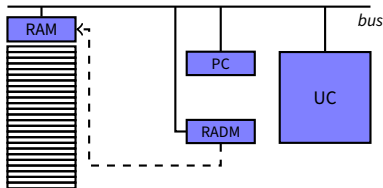
FONCTIONNEMENT DE LA MÉMOIRE

- L'**UC** va récupérer **les instructions** dans la mémoire **à partir de leur adresse** (*fetch*);



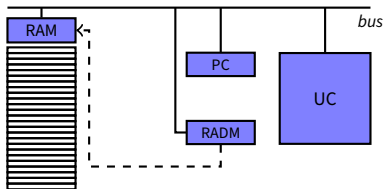
FONCTIONNEMENT DE LA MÉMOIRE

- L'**UC** va récupérer **les instructions** dans la mémoire **à partir de leur adresse** (*fetch*);
- L'**UC** va récupérer **les données** des variables dans la mémoire **à partir de leur adresse**;



FONCTIONNEMENT DE LA MÉMOIRE

- L'**UC** va récupérer **les instructions** dans la mémoire **à partir de leur adresse** (*fetch*);
- L'**UC** va récupérer **les données** des variables dans la mémoire **à partir de leur adresse**;
- L'**UC** écrit dans **des variables à une adresse donnée** dans la mémoire.



D'OÙ VIENNENT LES PROGRAMMES ?

D'OÙ VIENNENT LES PROGRAMMES ?

Le programme (code + données) est chargé depuis le disque vers la mémoire ...

D'OÙ VIENNENT LES PROGRAMMES ?

Le programme (code + données) est chargé depuis le disque vers la mémoire ... il est placé à un endroit donné dans la mémoire

D'OÙ VIENNENT LES PROGRAMMES ?

Le programme (code + données) est chargé depuis le disque vers la mémoire ... il est placé à un endroit donné dans la mémoire

Question

quelles sont les adresses des variables en mémoire ?

PROGRAMME VS PROCESSUS

Adresses **symboliques** vs Adresses **mémoires**

PROGRAMME VS PROCESSUS

Adresses **symboliques** vs Adresses **mémoires**

```
1 int a = 3;  
2 a = a + 2;
```

PROGRAMME VS PROCESSUS

Adresses **symboliques** vs Adresses **mémoires**

```
1 int a = 3;  
2 a = a + 2;
```

```
1 @a: memval 3  
2     mov eax, a  
3     mov ebx, 2  
4     add ecx, eax, ebx  
5     mov a, ecx
```

PROGRAMME VS PROCESSUS

Adresses **symboliques** vs Adresses **mémoires**

```
1 int a = 3;  
2 a = a + 2;
```

```
1 @a: memval 3  
2     mov eax, a  
3     mov ebx, 2  
4     add ecx, eax, ebx  
5     mov a, ecx
```

```
1 2B50: mov eax, 2B1E  
2 2B52: mov ebx, #0002  
3 2B54: add ecx, eax, ebx  
4 2B55: mov 2B1E, ecx  
5 ...  
6 2B1E: 0003
```

PROGRAMME VS PROCESSUS

Adresses **symboliques** vs Adresses **mémoires**

```
1 int a = 3;  
2 a = a + 2;
```

```
1 @a: memval 3  
2     mov eax, a  
3     mov ebx, 2  
4     add ecx, eax, ebx  
5     mov a, ecx
```

```
1 2B50: mov eax, 2B1E  
2 2B52: mov ebx, #0002  
3 2B54: add ecx, eax, ebx  
4 2B55: mov 2B1E, ecx  
5 ...  
6 2B1E: 0003
```

Édition de liens

- Lors de la création de processus, l'**OS instancie** le programme.
👉 transformer **les noms** des variables en **adresses**.

MÉTHODES DE LIAISON D'ADRESSES

MÉTHODES DE LIAISON D'ADRESSES

- À la compilation → on connaît les adresses des instructions et de toutes les données
 - 👉 adresses utilisées à l'intérieur d'un programme
 - 👉 adresses relatives au début du programme

MÉTHODES DE LIAISON D'ADRESSES

- **À la compilation** → on connaît les adresses des instructions et de toutes les données
 - 👉 adresses utilisées à l'intérieur d'un programme
 - 👉 adresses relatives au début du programme
- **Au chargement** → la taille du processus est fixée
 - 👉 **adresses virtuelles** : adresses utilisées dans le programme
 - 👉 **adresses physiques** : adresses utilisées dans la **RAM**

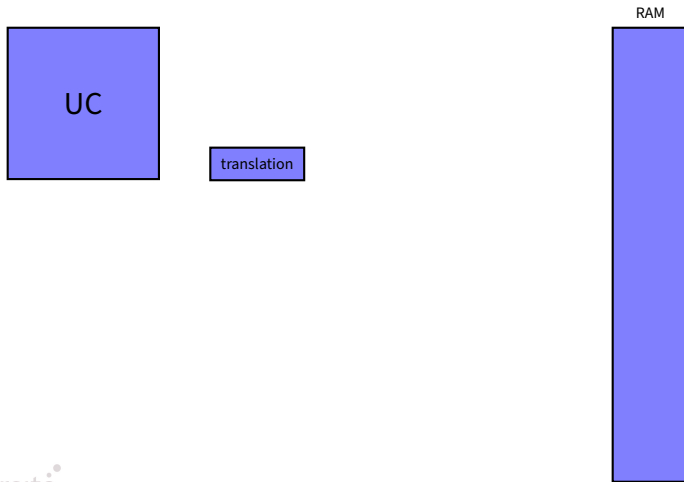
MÉTHODES DE LIAISON D'ADRESSES

- **À la compilation** → on connaît les adresses des instructions et de toutes les données
 - 👉 adresses utilisées à l'intérieur d'un programme
 - 👉 adresses relatives au début du programme
- **Au chargement** → la taille du processus est fixée
 - 👉 **adresses virtuelles** : adresses utilisées dans le programme
 - 👉 **adresses physiques** : adresses utilisées dans la **RAM**
 - 👉 nécessite un *composant de translation* (**MMU**) dans l'**UC**

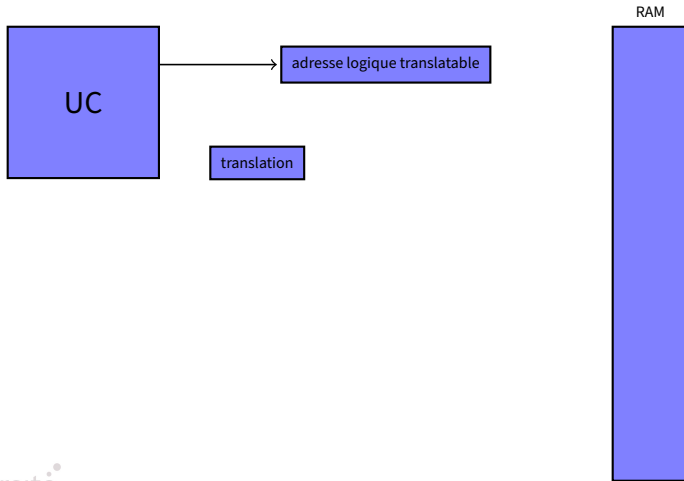
MÉTHODES DE LIAISON D'ADRESSES

- **À la compilation** → on connaît les adresses des instructions et de toutes les données
 - 👉 adresses utilisées à l'intérieur d'un programme
 - 👉 adresses relatives au début du programme
- **Au chargement** → la taille du processus est fixée
 - 👉 **adresses virtuelles** : adresses utilisées dans le programme
 - 👉 **adresses physiques** : adresses utilisées dans la **RAM**
 - 👉 nécessite un *composant de translation* (**MMU**) dans l'**UC**
- **À l'exécution** → le processus a besoin de plus de place
 - 👉 déplacer le processus + réédition de lien

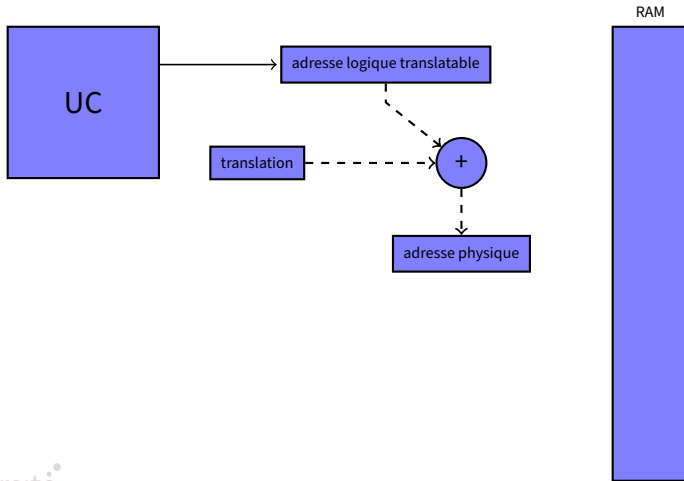
RÉSOLUTION D'ADRESSE



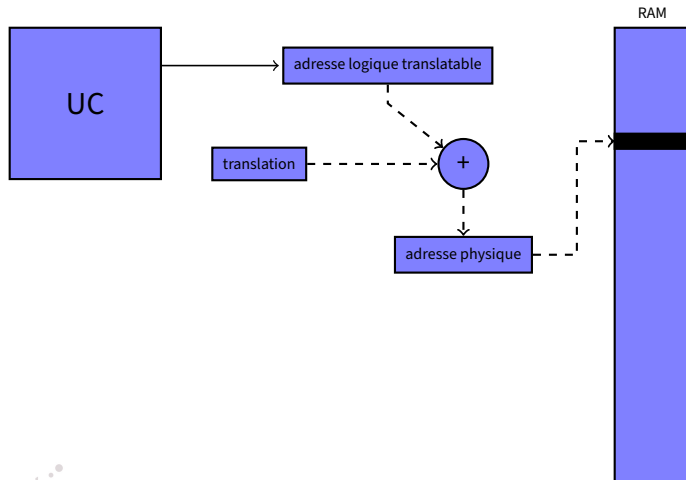
RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



STRATÉGIES D'ALLOCATION DE MÉMOIRE

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles :

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles :
 1. **Allocation contiguë** de cases mémoire (**par partition**)

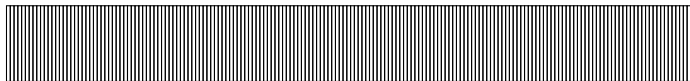
STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles :
 1. **Allocation contiguë** de cases mémoire (**par partition**)
 2. **Allocation non contiguë** (**par pagination**)

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

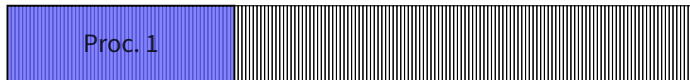
RAM



ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

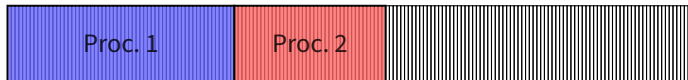
RAM



ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

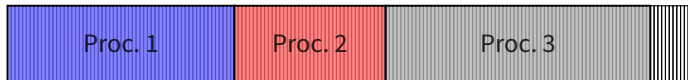
RAM



ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

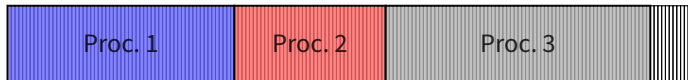
RAM



ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

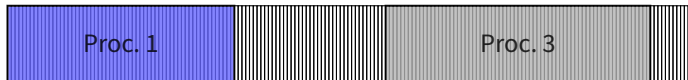
RAM



ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM

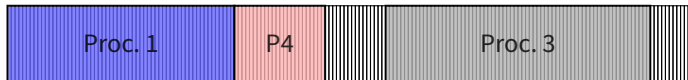


✗ des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM

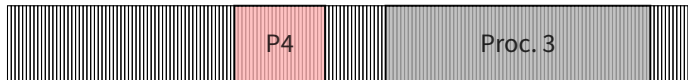


✗ des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM

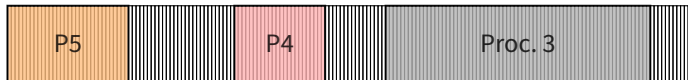


✗ des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

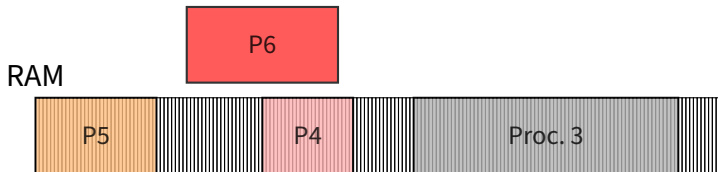
RAM



✗ des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

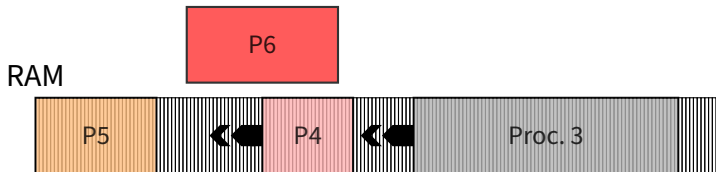
Les processus constituent **un seul bloc** non décomposable.



- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer

ALLOCATION PAR PARTITION

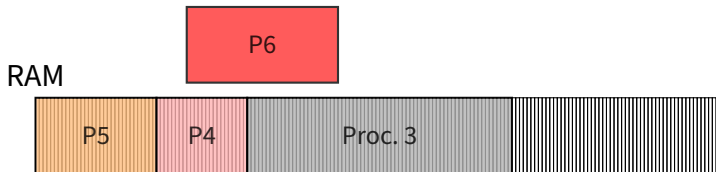
Les processus constituent **un seul bloc** non décomposable.



- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

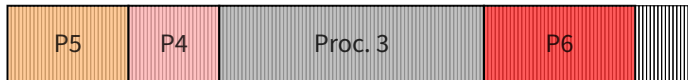


- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM

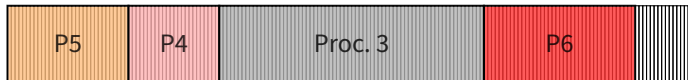


- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM



- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer → **défragmenter**

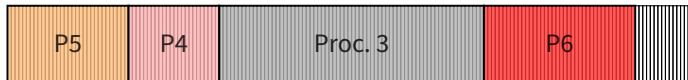


réduire la **fragmentation**

ALLOCATION PAR PARTITION

Les processus constituent **un seul bloc** non décomposable.

RAM



- ✗ des trous apparaissent → **fragmentation**
- ✗ les gros processus ne peuvent pas rentrer → **défragmenter**



réduire la **fragmentation**



limiter les opérations de **défragmentation**

ALLOCATION PAR PARTITION

- **Stratégies d'allocation** → choisir dans quelle *zone libre* placer un processus

ALLOCATION PAR PARTITION

- **Stratégies d'allocation** → choisir dans quelle *zone libre* placer un processus

👉 **First Fit** : premier bloc libre

ALLOCATION PAR PARTITION

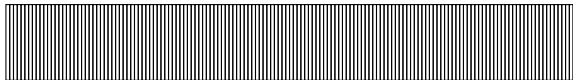
- **Stratégies d'allocation** → choisir dans quelle *zone libre* placer un processus
 - 👉 **First Fit** : premier bloc libre
 - 👉 **Best Fit** : plus petit bloc libre

ALLOCATION PAR PARTITION

- **Stratégies d'allocation** → choisir dans quelle *zone libre* placer un processus
 - 👉 **First Fit** : premier bloc libre
 - 👉 **Best Fit** : plus petit bloc libre
 - 👉 **Worst Fit** : plus grand bloc libre

ALLOCATION PAR PAGINATION

RAM



Processus :

code, bibliothèques
environnement, tas, pile...

ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**

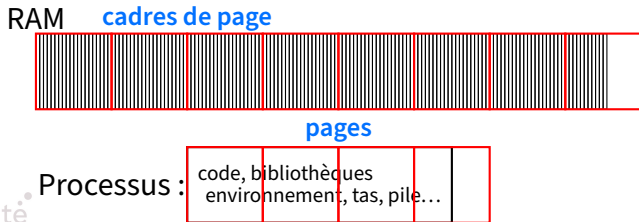


Processus :

code, bibliothèques
environnement, tas, pile...

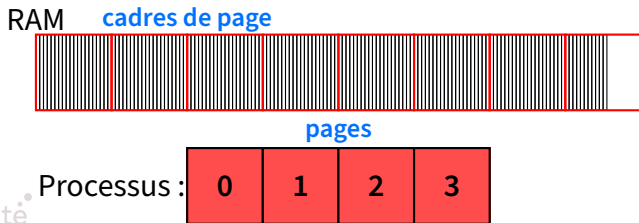
ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c



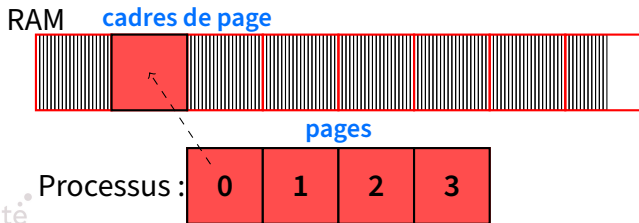
ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c
 - chaque **page** a la **même taille** qu'un **bloc**



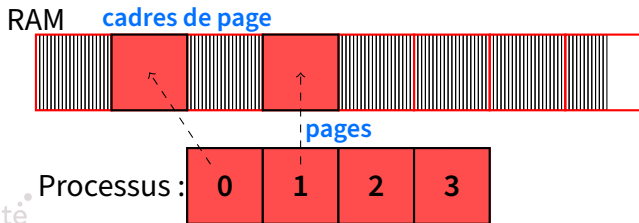
ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c
 - chaque **page** a la **même taille** qu'un **bloc**
- Placer les pages dans les cadres



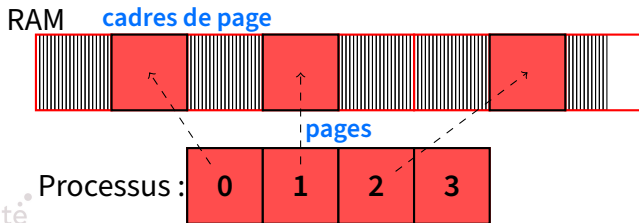
ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c
 - chaque **page** a la **même taille** qu'un **bloc**
- Placer les pages dans les cadres



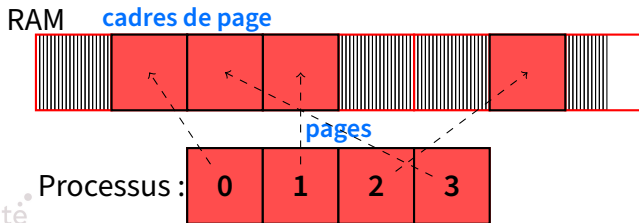
ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c
 - chaque **page** a la **même taille** qu'un **bloc**
- Placer les pages dans les cadres



ALLOCATION PAR PAGINATION

- Découper la **mémoire physique** en **blocs** de taille T_c constante, appelés **cadres de pages**
- Découper l'**espace mémoire** utilisé par un processus (**espace logique**) en paquets de x **pages** de taille T_c
 - chaque **page** a la **même taille** qu'un **bloc**
- Placer les pages dans les cadres



ALLOCATION PAR PAGINATION

ALLOCATION PAR PAGINATION

- Allocation mémoire :

- 👉 un processus est dans des zones *disjointes*
- 👉 pas besoin de défragmenter

ALLOCATION PAR PAGINATION

- Allocation mémoire :

- 👉 un processus est dans des zones *disjointes*
- 👉 pas besoin de défragmenter

- Adaptation :

- 👉 besoin de plus de mémoire → *rajouter* des pages
- 👉 pas besoin de le ré-allouer entièrement

ALLOCATION PAR PAGINATION

- **Allocation mémoire :**
 - 👉 un processus est dans des zones *disjointes*
 - 👉 pas besoin de défragmenter
- **Adaptation :**
 - 👉 besoin de plus de mémoire → *rajouter* des pages
 - 👉 pas besoin de le ré-allouer entièrement
- **Mémoire virtuelle :** charger uniquement les pages dont le processus a besoin.

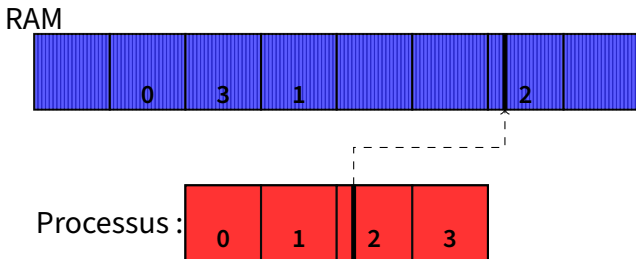
ADRESSAGE

ADRESSAGE

- Déterminer l'adresse **physique** à partir de l'adresse **logique**

ADRESSAGE

- Déterminer l'adresse **physique** à partir de l'adresse **logique**



ADRESSAGE

- Déterminer l'adresse **physique** à partir de l'adresse **logique**
- **Adresse logique**
 - Numéro de page (n bits) + décalage (m bits)

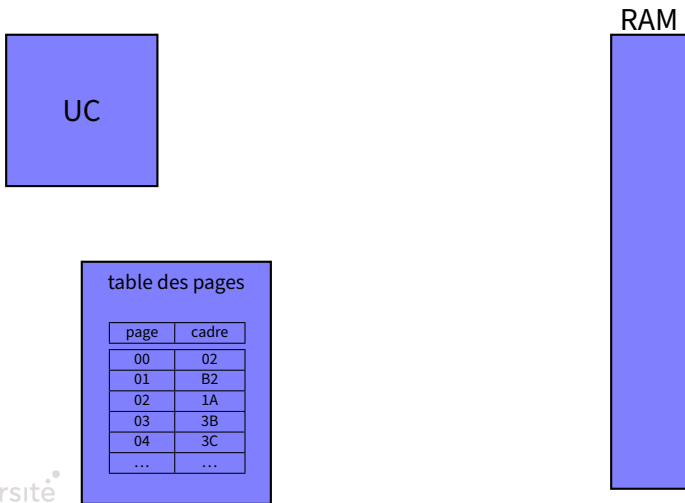
ADRESSAGE

- Déterminer l'adresse **physique** à partir de l'adresse **logique**
- **Adresse logique**
 - Numéro de page (n bits) + décalage (m bits)
- Chaque processus maintient une liste :
 - numéro de page \rightarrow numéro de cadre

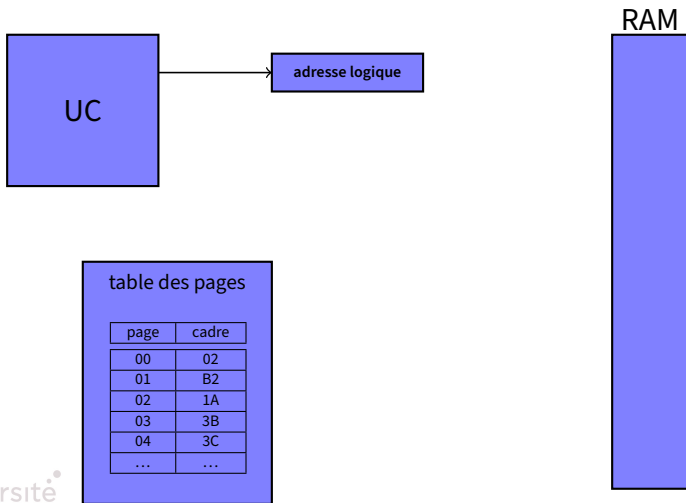
ADRESSAGE

- Déterminer l'adresse **physique** à partir de l'adresse **logique**
- **Adresse logique**
 - Numéro de page (n bits) + décalage (m bits)
- Chaque processus maintient une liste :
 - numéro de page \rightarrow numéro de cadre
 - c'est la **table des pages**

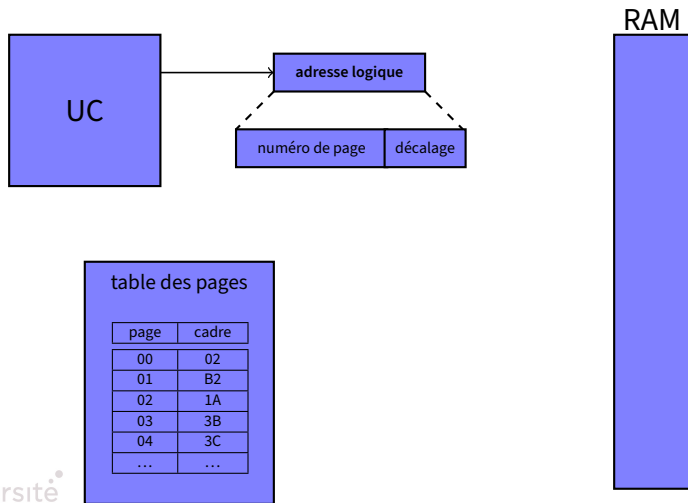
RÉSOLUTION D'ADRESSE



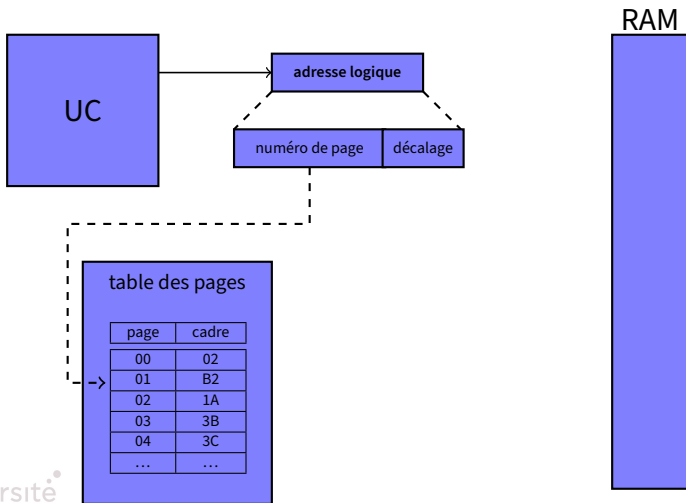
RÉSOLUTION D'ADRESSE



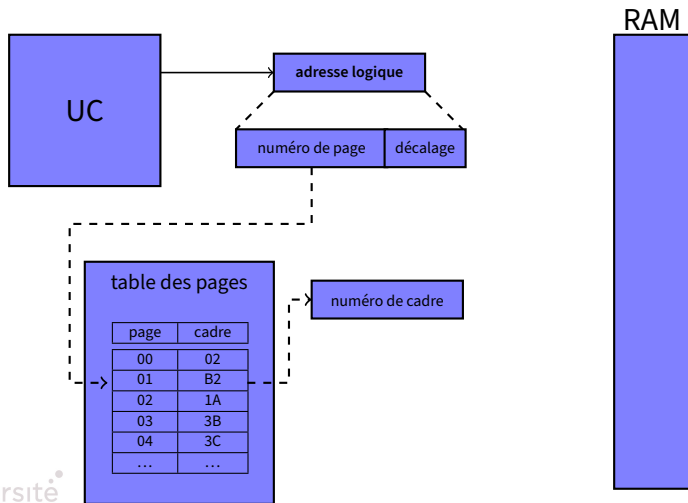
RÉSOLUTION D'ADRESSE



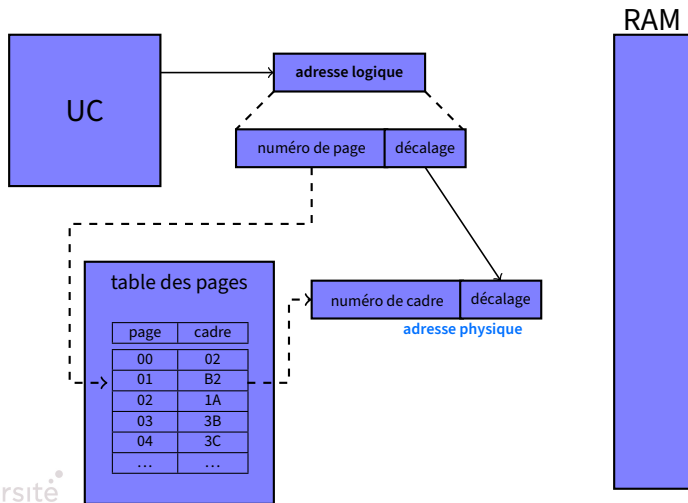
RÉSOLUTION D'ADRESSE



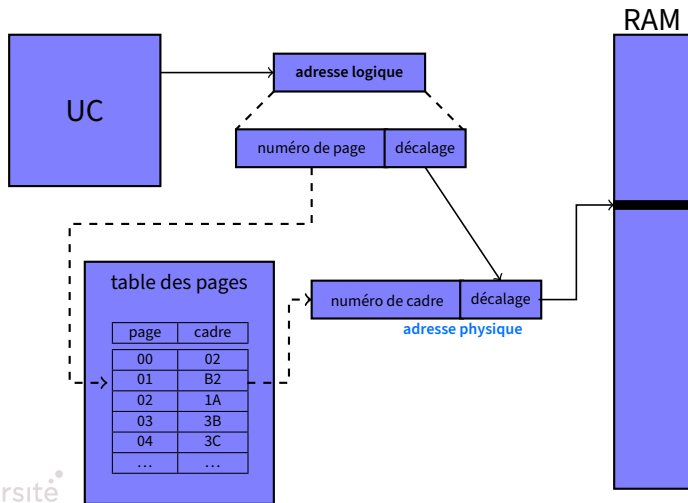
RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**
 - L'OS charge la table des pages du processus dans la MMU

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**
 - L'OS charge la table des pages du processus dans la MMU
 - Pas de calcul d'adresse au niveau de l'OS

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**
 - L'OS charge la table des pages du processus dans la MMU
 - Pas de calcul d'adresse au niveau de l'OS
- **Allocation des cadres** : ne pas allouer le même cadre à deux processus différents

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**
 - L'OS charge la table des pages du processus dans la MMU
 - Pas de calcul d'adresse au niveau de l'OS
- **Allocation des cadres** : ne pas allouer le même cadre à deux processus différents
 - 👉 L'OS doit savoir quel processus utilise quel cadre

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel → la **MMU**
 - L'OS charge la table des pages du processus dans la MMU
 - Pas de calcul d'adresse au niveau de l'OS
- **Allocation des cadres** : ne pas allouer le même cadre à deux processus différents
 - 👉 L'OS doit savoir quel processus utilise quel cadre
 - 👉 **Table des cadres de page libres**

num. cadre	num. proc.	libre
0	42	1
1	37	1
2	-	0
...

GESTION DE L'ESPACE D'ADRESSAGE

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus
 - Allocation et commutation plus coûteuse en temps

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus
 - Allocation et commutation plus coûteuse en temps
- *Pages trop grosses* (ex : $n = 10, m = 22$)

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus
 - Allocation et commutation plus coûteuse en temps
- *Pages trop grosses* (ex : $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus
 - Allocation et commutation plus coûteuse en temps
- *Pages trop grosses* (ex : $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue
 - 2^{21} octets ≈ 2 Mo par processus

GESTION DE L'ESPACE D'ADRESSAGE

Problème → Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits $\approx 2,5$ Mo par processus
 - Allocation et commutation plus coûteuse en temps
- *Pages trop grosses* (ex : $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue
 - 2^{21} octets ≈ 2 Mo par processus
 - Pas gérable au niveau du MMU

PAGINATION HIÉRARCHIQUE 2 NIVEAUX (OU PLUS)

PAGINATION HIÉRARCHIQUE 2 NIVEAUX (OU PLUS)

- **Solution** → paginer la table des pages

PAGINATION HIÉRARCHIQUE

2 NIVEAUX (OU PLUS)

- **Solution** → paginer la table des pages
 - 👉 ne charger que les tables utiles

PAGINATION HIÉRARCHIQUE

2 NIVEAUX (OU PLUS)

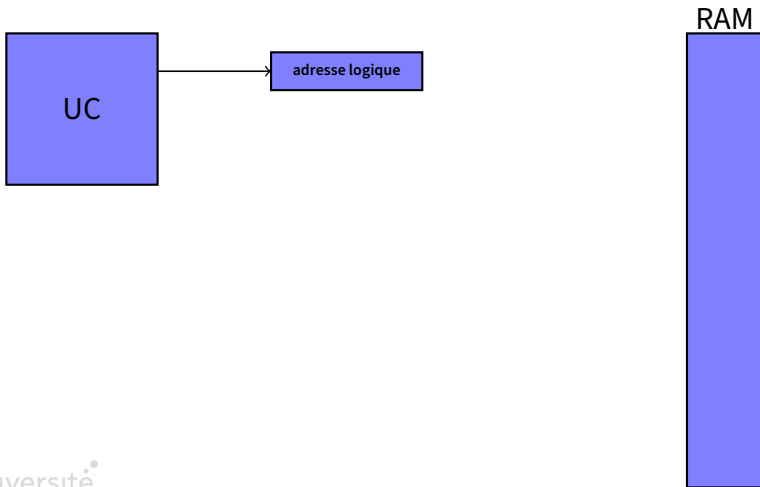
- **Solution** → paginer la table des pages
 - 👉 ne charger que les tables utiles
 - 👉 réduire l'espace mémoire utilisé par le système d'adressage

PAGINATION HIÉRARCHIQUE

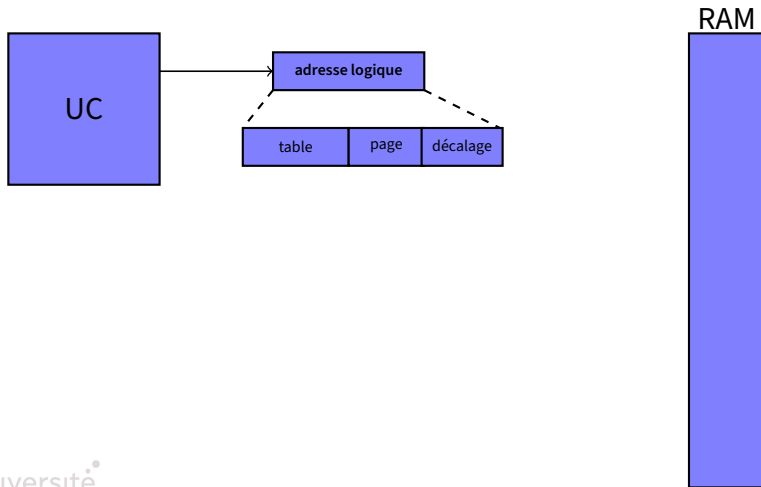
2 NIVEAUX (OU PLUS)

- **Solution** → paginer la table des pages
 - 👉 ne charger que les tables utiles
 - 👉 réduire l'espace mémoire utilisé par le système d'adressage
 - 👉 réduire la fragmentation due aux pages

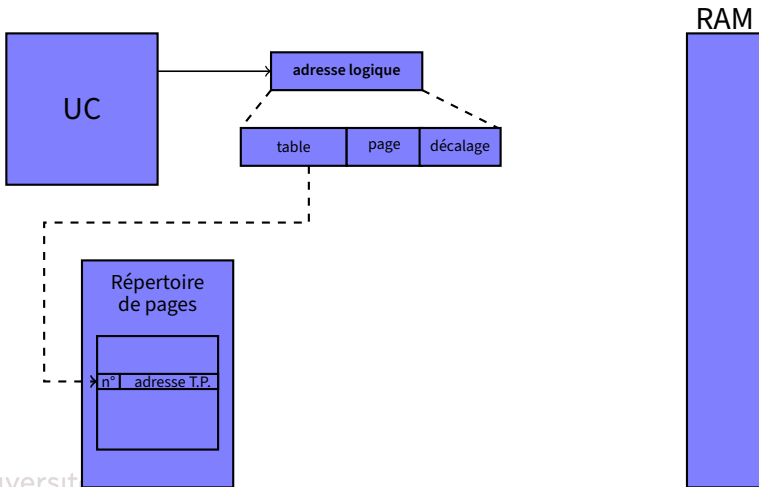
PAGINATION À DEUX NIVEAUX



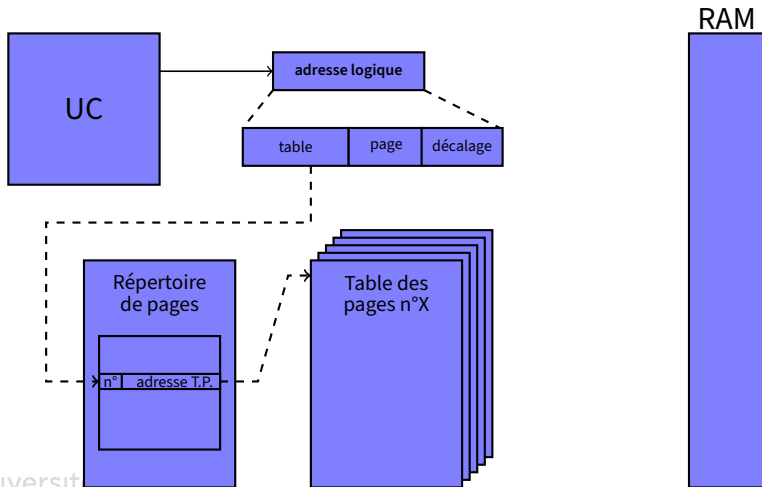
PAGINATION À DEUX NIVEAUX



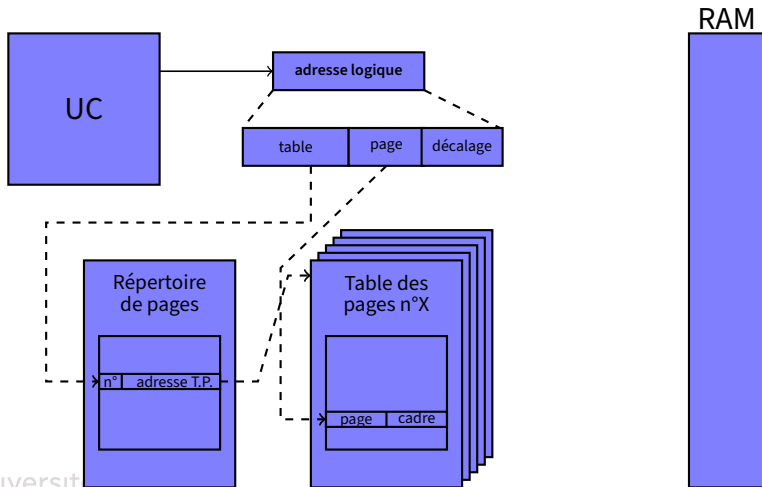
PAGINATION À DEUX NIVEAUX



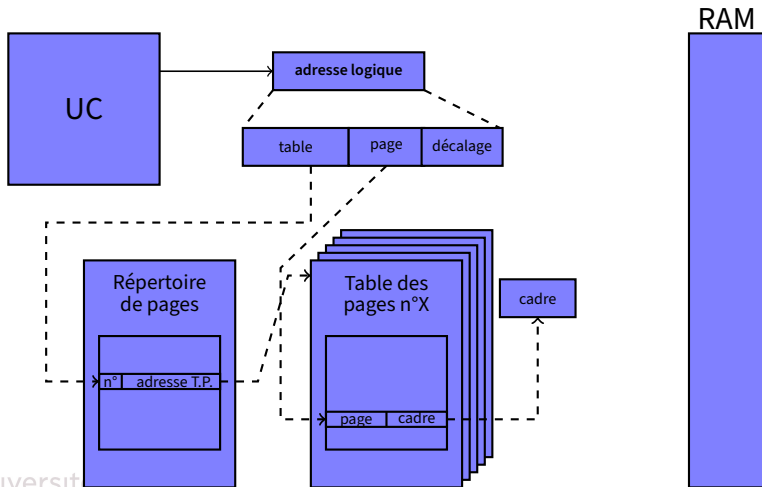
PAGINATION À DEUX NIVEAUX



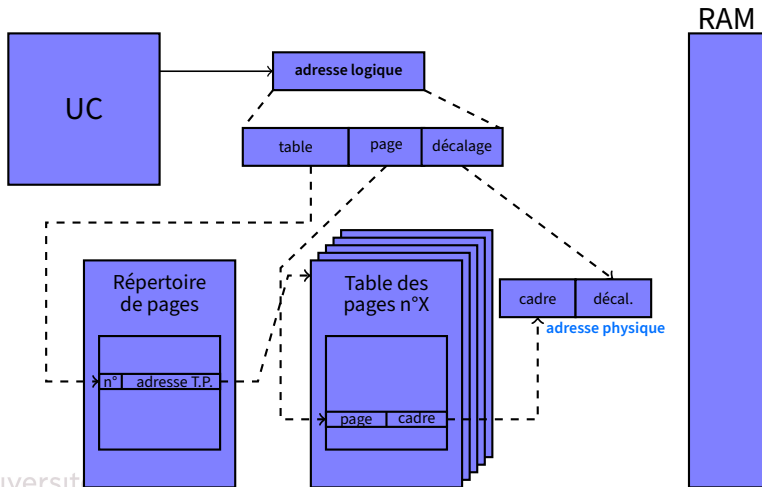
PAGINATION À DEUX NIVEAUX



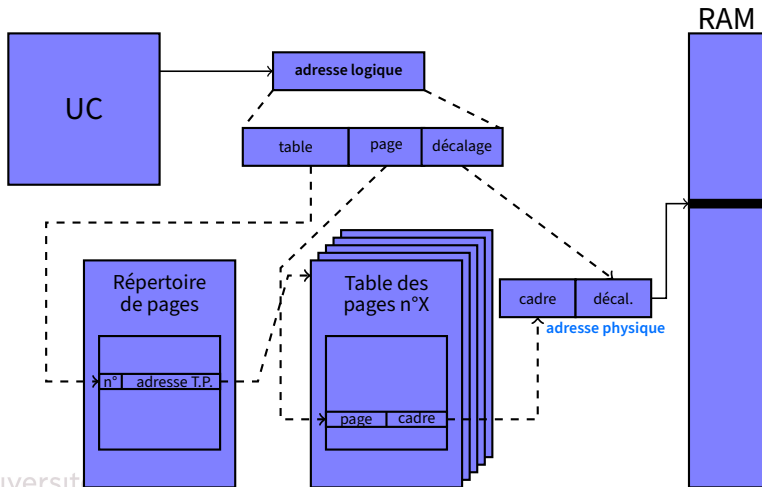
PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



EXEMPLE

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits
 - Total = $2^{20} \times 20 \approx 2,5$ Mo par processus

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits
 - Total = $2^{20} \times 20 \approx 2,5$ Mo par processus
- Pagination à 2 niveaux ($n_1 = 10, n_2 = 10, m = 12$)

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits
 - Total = $2^{20} \times 20 \approx 2,5$ Mo par processus
- Pagination à 2 niveaux ($n_1 = 10, n_2 = 10, m = 12$)
 - Répertoire = 2^{10} lignes de 32 bits ≈ 4 Ko

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits
 - Total = $2^{20} \times 20 \approx 2,5$ Mo par processus
- Pagination à 2 niveaux ($n_1 = 10, n_2 = 10, m = 12$)
 - Répertoire = 2^{10} lignes de 32 bits ≈ 4 Ko
 - 1 table de pages = 2^{10} lignes de 20 bits $\approx 2,5$ Ko

EXEMPLE

- Adresses sur 32 bits et cadres de 4 Ko $\rightarrow m = 12$
- Pagination à 1 niveau ($n = 20, m = 12$)
 - Table des pages = 2^n lignes de n bits
 - Total = $2^{20} \times 20 \approx 2,5$ Mo par processus
- Pagination à 2 niveaux ($n_1 = 10, n_2 = 10, m = 12$)
 - Répertoire = 2^{10} lignes de 32 bits ≈ 4 Ko
 - 1 table de pages = 2^{10} lignes de 20 bits $\approx 2,5$ Ko
 - Total = entre 6,5 Ko et 2,5 Mo par processus

PLAN

- La mémoire
- La mémoire virtuelle
- La mémoire segmentée
- Synthèse

[Retour à l'accueil](#) - [Retour au plan](#)

ÉTAT DE LA MÉMOIRE

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées
 - Traitement d'erreur \rightarrow rarement utilisé

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées
 - Traitement d'erreur \rightarrow rarement utilisé
 - Données (tableau, jeu, ...) \rightarrow pas tout en même temps

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées
 - Traitement d'erreur \rightarrow rarement utilisé
 - Données (tableau, jeu, ...) \rightarrow pas tout en même temps
 - Bibliothèque \rightarrow très variable!

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées
 - Traitement d'erreur \rightarrow rarement utilisé
 - Données (tableau, jeu, ...) \rightarrow pas tout en même temps
 - Bibliothèque \rightarrow très variable!

👉 ne charger que les pages utiles!

ÉTAT DE LA MÉMOIRE

- Nombre et taille des processus :
 - Entre 200 et 500 processus en parallèle sur un PC
 - Gros processus : Eclipse = 250 Mo, données d'un jeu ≥ 1 Go
 - ✗ Somme des tailles des processus \geq Capacité RAM
- Portions de code inutilisées
 - Traitement d'erreur \rightarrow rarement utilisé
 - Données (tableau, jeu, ...) \rightarrow pas tout en même temps
 - Bibliothèque \rightarrow très variable!

👉 ne charger que les pages utiles!



laisser le reste sur le disque

MÉMOIRE VIRTUELLE

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (**swap**) → la RAM est un cache

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (**swap**) → la RAM est un cache
- Chaque ligne de **la table des pages** contient :

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (**swap**) → la RAM est un cache
- Chaque ligne de **la table des pages** contient :
 - un **bit de la validité** qui indique si la page est en RAM

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (**swap**) → la RAM est un cache
- Chaque ligne de **la table des pages** contient :
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)

MÉMOIRE VIRTUELLE

- *Extension* des mécanismes de **pagination**
 - la mémoire paginée peut représenter des **espaces non présents en RAM** (ex. disque) → **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (**swap**) → la RAM est un cache
- Chaque ligne de **la table des pages** contient :
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)
 - sinon une information pour la trouver sur **le disque**

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace
qu'il n'a effectivement en **mémoire physique**

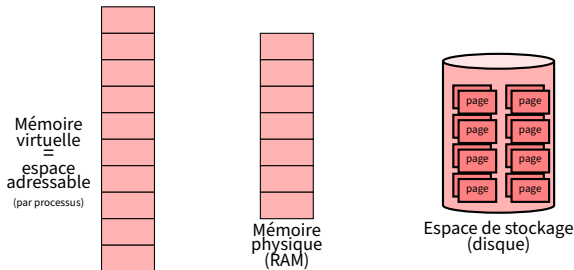


SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

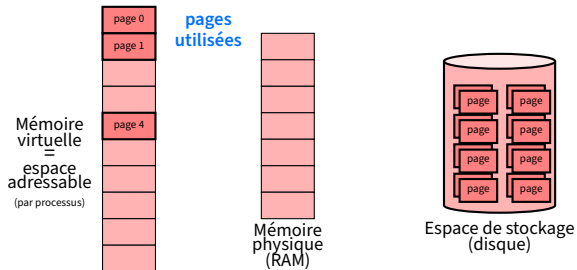


SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

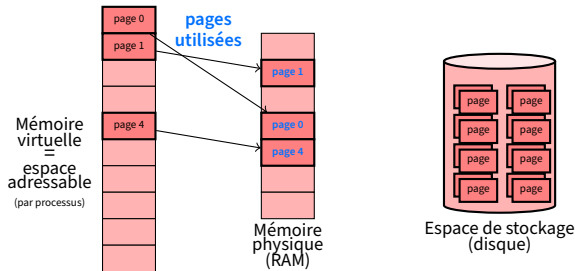


SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

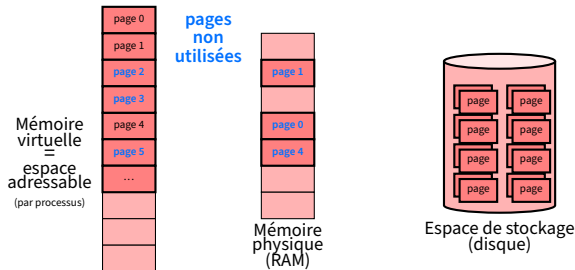


SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

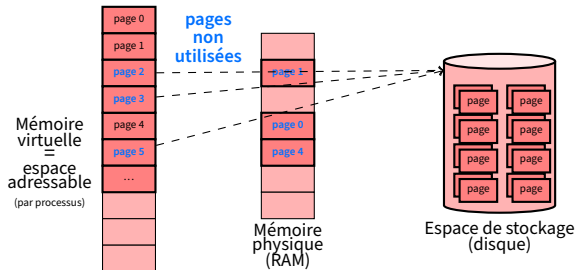


SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace
qu'il n'a effectivement en **mémoire physique**

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace
qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :
👉 levée d'une **exception** CPU **Page-Fault**

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace
qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM
- **Avantages**

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM
- **Avantages**
 - ✓ masquer la taille de la RAM

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM
- **Avantages**
 - ✓ masquer la taille de la RAM
 - ✓ possibilité de mettre plus de processus en parallèles

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

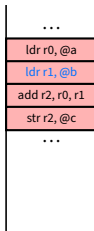
- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM
- **Avantages**
 - ✓ masquer la taille de la RAM
 - ✓ possibilité de mettre plus de processus en parallèles
 - ✓ affecter plusieurs adresses virtuelles à une adresse physique

SCHÉMA DE PRINCIPE

Chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

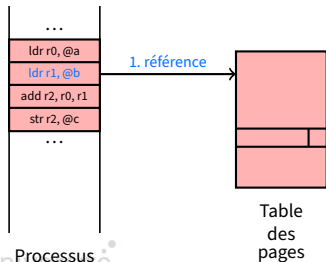
- Accès à une page non présente en RAM :
 - 👉 levée d'une **exception** CPU **Page-Fault**
 - 👉 *processus courant bloqué* et **chargement** de la page en RAM
- **Avantages**
 - ✓ masquer la taille de la RAM
 - ✓ possibilité de mettre plus de processus en parallèles
 - ✓ affecter plusieurs adresses virtuelles à une adresse physique
 - ✓ une pagination à la demande

PAGINATION À LA DEMANDE

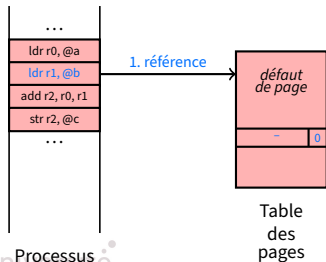


Processus

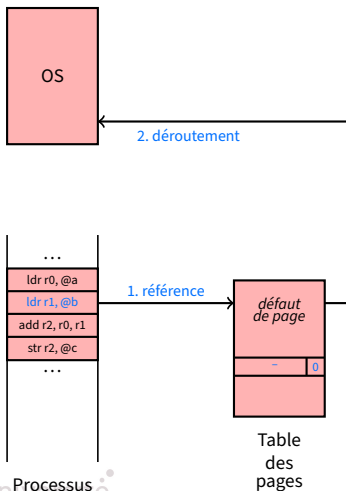
PAGINATION À LA DEMANDE



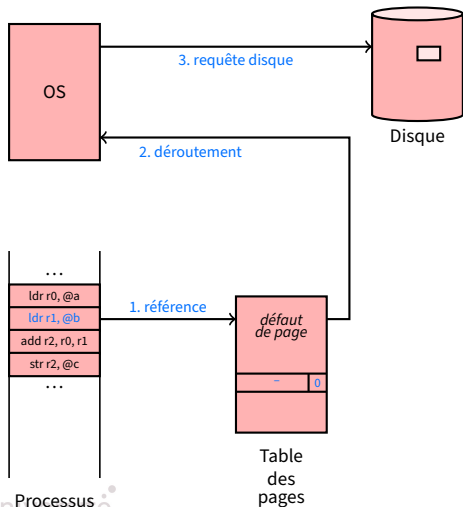
PAGINATION À LA DEMANDE



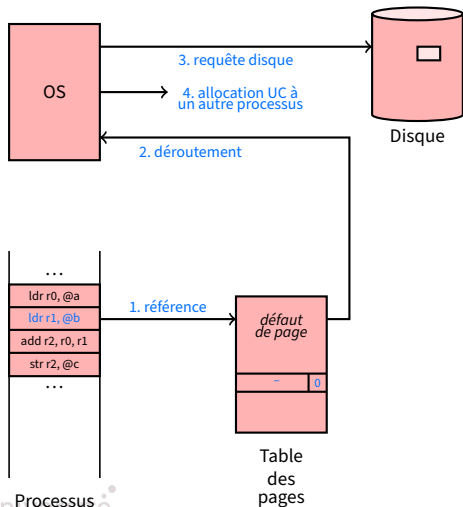
PAGINATION À LA DEMANDE



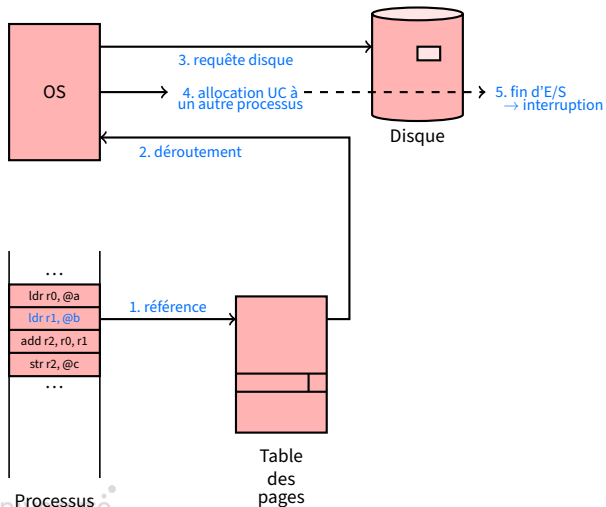
PAGINATION À LA DEMANDE



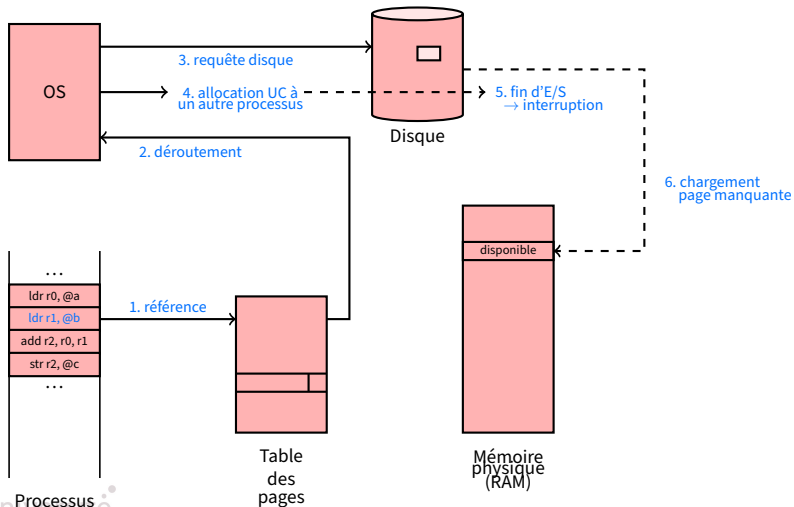
PAGINATION À LA DEMANDE



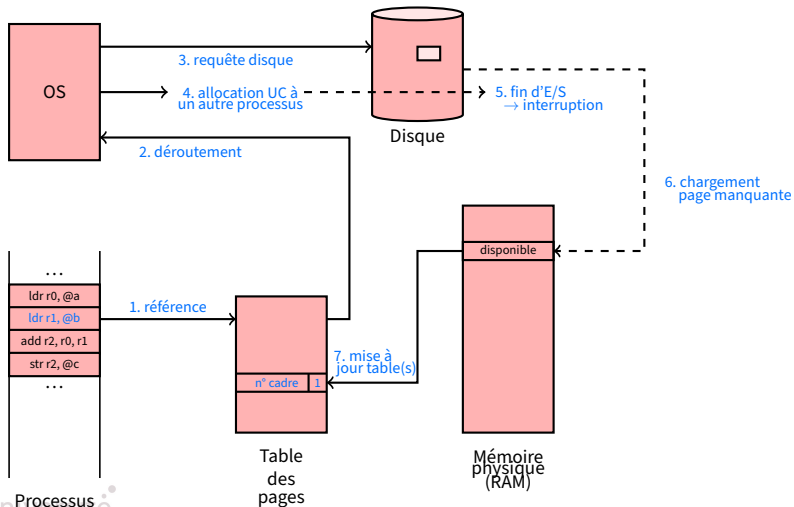
PAGINATION À LA DEMANDE



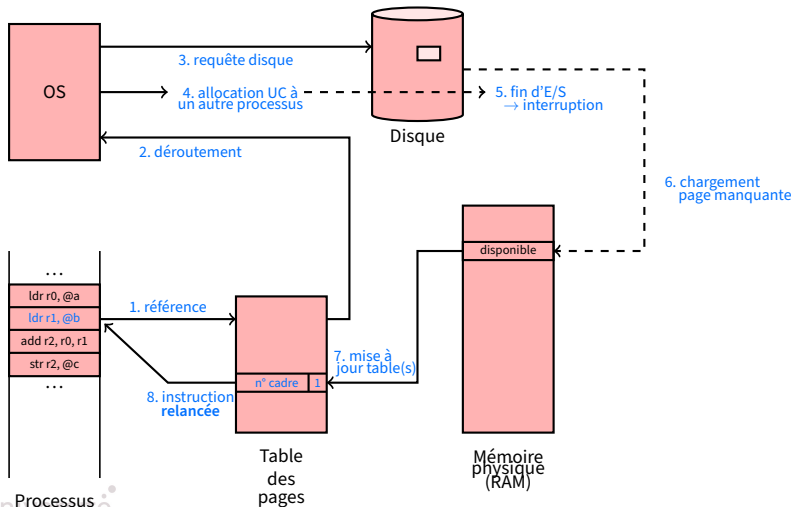
PAGINATION À LA DEMANDE



PAGINATION À LA DEMANDE



PAGINATION À LA DEMANDE



COÛT DES DÉFAUTS DE PAGE

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire

COÛT DES DÉFAUTS DE PAGE

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

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$
👉 dépend à la probabilité d'avoir un défaut de page

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$
 - 👉 dépend à la probabilité d'avoir un défaut de page
 - 👉 en pratique, M est 1000 fois plus petit que D .

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$
 - 👉 dépend à la probabilité d'avoir un défaut de page
 - 👉 en pratique, M est 1000 fois plus petit que D .
- Temps d'exécution proportionnel à la probabilité de défaut de page

COÛT DES DÉFAUTS DE PAGE

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$
 - 👉 dépend à la probabilité d'avoir un défaut de page
 - 👉 en pratique, M est 1000 fois plus petit que D .
- Temps d'exécution proportionnel à la probabilité de défaut de page
 - 👉 réduire le nombre de défauts de page

ALLOCATION DES PAGES

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
 - moins de processus → **ralentissement**

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**
 - libérer un cadre lorsqu'on a besoin d'une nouvelle page

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**
 - libérer un cadre lorsqu'on a besoin d'une nouvelle page
- Politiques d'allocation

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**
 - libérer un cadre lorsqu'on a besoin d'une nouvelle page
- Politiques d'allocation
 - 👉 Allocation équitable

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**
 - libérer un cadre lorsqu'on a besoin d'une nouvelle page
- Politiques d'allocation
 - 👉 Allocation équitable
 - 👉 Allocation proportionnelle

ALLOCATION DES PAGES

- Combien de cadres (RAM) alloués à chaque processus?
 - un nombre de cadres limité (selon la politique d'allocation)
 - 👉 plus de cadre par processus
→ moins de processus → **ralentissement**
 - 👉 moins de cadre par processus
→ plus de défauts → **ralentissement**
 - libérer un cadre lorsqu'on a besoin d'une nouvelle page
- Politiques d'allocation
 - 👉 Allocation équitable
 - 👉 Allocation proportionnelle
 - 👉 Allocation basée sur la priorité

ALLOCATION ÉQUITABLE

ALLOCATION ÉQUITABLE

- Principe

ALLOCATION ÉQUITABLE

- Principe
 - N cadres disponibles en RAM

ALLOCATION ÉQUITABLE

- Principe
 - N cadres disponibles en RAM
 - P processus

ALLOCATION ÉQUITABLE

- Principe

- N cadres disponibles en RAM
- P processus
- Chaque processus reçoit N/P cadres

ALLOCATION ÉQUITABLE

- Principe
 - N cadres disponibles en RAM
 - P processus
 - Chaque processus reçoit N/P cadres
 - le reste sert de tampon

ALLOCATION ÉQUITABLE

- **Principe**

- N cadres disponibles en RAM
- P processus
- Chaque processus reçoit N/P cadres
 - le reste sert de tampon

- **Inconvénient** → tous les processus n'ont pas besoin de la même quantité de mémoire...

ALLOCATION PROPORTIONNELLE

- Principe
 - N cadres disponibles en RAM
 - P processus

ALLOCATION PROPORTIONNELLE

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages

ALLOCATION PROPORTIONNELLE

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, M_i la taille de P_i

ALLOCATION PROPORTIONNELLE

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, M_i la taille de P_i
- P_i reçoit $(N - T) \times \frac{M_i}{\sum_i M_i}$ cadres

ALLOCATION PROPORTIONNELLE

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, M_i la taille de P_i
- P_i reçoit $(N - T) \times \frac{M_i}{\sum_i M_i}$ cadres

- **Inconvénient** → les petits processus font plus de défauts de pages

ALLOCATION BASÉE SUR LA PRIORITÉ

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages

ALLOCATION BASÉE SUR LA PRIORITÉ

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, $prio_i$ la priorité de P_i

ALLOCATION BASÉE SUR LA PRIORITÉ

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, $prio_i$ la priorité de P_i
- P_i reçoit $(N - T) \times \frac{prio_i}{\sum_i prio_i}$

ALLOCATION BASÉE SUR LA PRIORITÉ

- Principe

- N cadres disponibles en RAM
- P processus
- On garde généralement un tampon de T pages
- $\forall i \in [1, P]$, $prio_i$ la priorité de P_i
- P_i reçoit $(N - T) \times \frac{prio_i}{\sum_i prio_i}$

- **Inconvénient** → les processus moins prioritaires font plus de défauts de pages

REEMPLACEMENT DE PAGES

- p = probabilité d'avoir un défaut de page
- M = temps d'accès à la mémoire
- D = temps de traitement du défaut de page
- Temps d'accès = $(1 - p) \times M + p \times D = M + p \times (D - M)$
 - 👉 dépend à la probabilité d'avoir un défaut de page
 - 👉 en pratique, M est 1000 fois plus petit que D .
- Temps d'exécution proportionnel à la probabilité de défaut de page
 - 👉 réduire le nombre de défauts de page

REEMPLACEMENT DE PAGES

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04

--	--	--	--

Défauts : 0

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04

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

Défauts : 4

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04 2A

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

Défauts : 4

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04 2A

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

Défauts : 4

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04 2A 12

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

Défauts : 4

REEMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04 2A 12

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

Défauts : 5

REEMPLACEMENT DE PAGES

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

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

Défauts : 5

REEMPLACEMENT DE PAGES

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

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

Défauts : 6

REPRÉSENTATION VISUELLE

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

REPRÉSENTATION VISUELLE

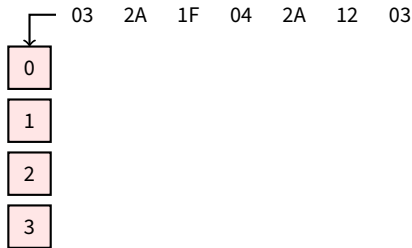
Figurer l'état du cache dans le temps
pour compter le nombre de défauts

03 2A 1F 04 2A 12 03

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

REPRÉSENTATION VISUELLE

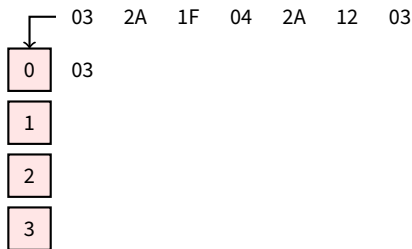
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une ligne par cadre de page alloué au processus

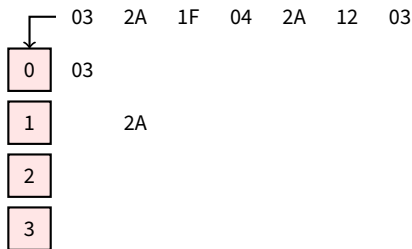
REPRÉSENTATION VISUELLE

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



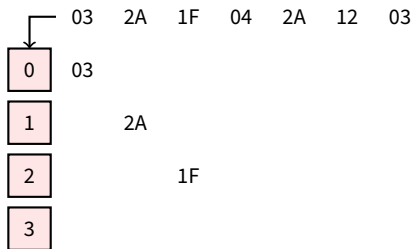
REPRÉSENTATION VISUELLE

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



REPRÉSENTATION VISUELLE

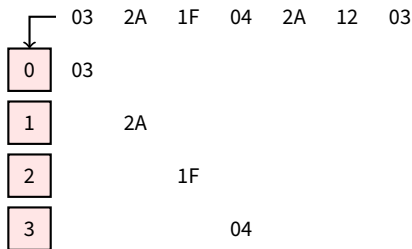
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

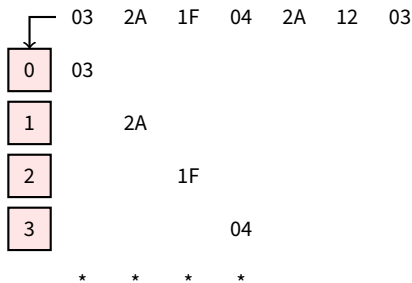
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

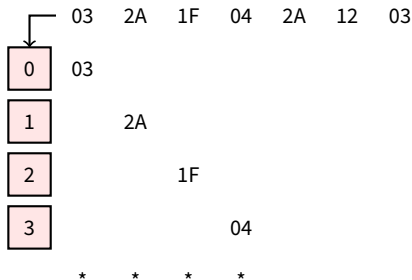
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Marquer les défauts au fur et à mesure

REPRÉSENTATION VISUELLE

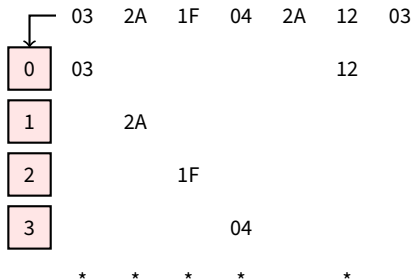
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Pas de défaut → laisser en blanc

REPRÉSENTATION VISUELLE

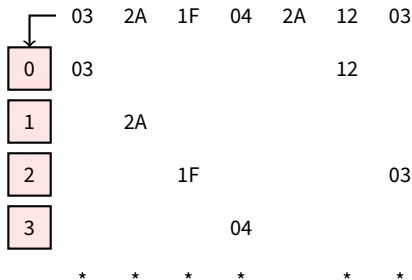
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Et on continue...

REPRÉSENTATION VISUELLE

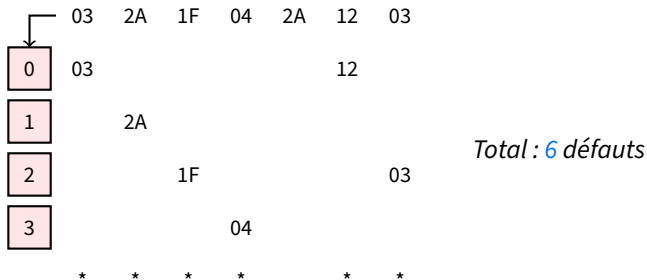
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Et on continue...

REPRÉSENTATION VISUELLE

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



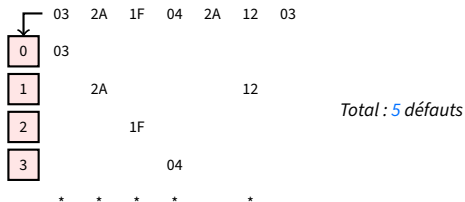
Et on continue...

SOLUTION OPTIMALE

Il existe une politique de remplacement optimale...
...si on connaît à l'avance les références aux pages!

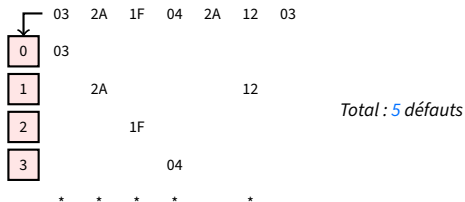
SOLUTION OPTIMALE

Il existe une politique de remplacement optimale...
...si on connaît à l'avance les références aux pages!



SOLUTION OPTIMALE

Il existe une politique de remplacement optimale...
...si on connaît à l'avance les références aux pages!



En pratique, on ne connaît pas les références!

ALGORITHME DE REMPLACEMENT

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes
 - **First In, First Out (FIFO)**

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes
 - **First In, First Out (FIFO)**
 - 👉 retirer les pages les plus anciennes

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes
 - **First In, First Out (FIFO)**
 - 👉 retirer les pages les plus anciennes
 - 👉 seconde chance

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes
 - **First In, First Out (FIFO)**
 - 👉 retirer les pages les plus anciennes
 - 👉 seconde chance
 - **Basés sur l'utilisation des pages**

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes
 - **First In, First Out (FIFO)**
 - 👉 retirer les pages les plus anciennes
 - 👉 seconde chance
 - **Basés sur l'utilisation des pages**
 - 👉 moins utilisée (**Least Frequently Used**)

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes

- **First In, First Out (FIFO)**

- 👉 retirer les pages les plus anciennes
 - 👉 seconde chance

- **Basés sur l'utilisation des pages**

- 👉 moins utilisée (**Least Frequently Used**)
 - 👉 pas récemment utilisée (**Not Recently Used**)

ALGORITHME DE REMPLACEMENT

Définir **une fonction** qui, étant donné l'état actuel (occupation des cadres), *décide quel cadre libérer*.

- Les classes d'algorithmes

- **First In, First Out (FIFO)**

- 👉 retirer les pages les plus anciennes
 - 👉 seconde chance

- **Basés sur l'utilisation des pages**

- 👉 moins utilisée (**Least Frequently Used**)
 - 👉 pas récemment utilisée (**Not Recently Used**)
 - 👉 moins récemment utilisée (**Least Recently Used**)

ALGORITHMES TYPE FIFO

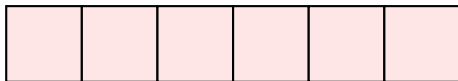
Retirer la page la plus ancienne

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

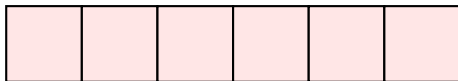


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

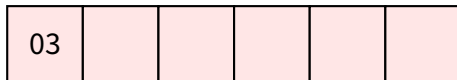


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

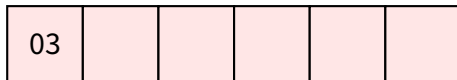


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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



ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

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

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une implémentation de l'algorithme

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

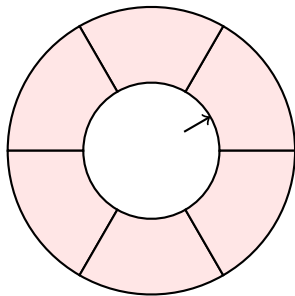
37	B1	03	76	2A	1F
----	----	----	----	----	----

ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

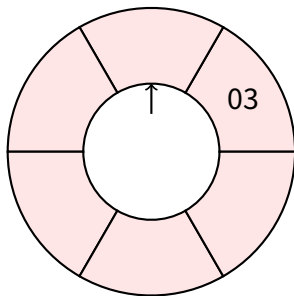


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

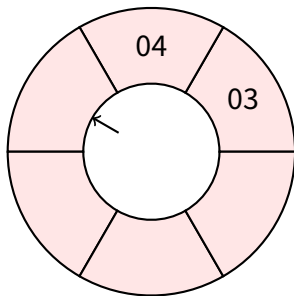


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

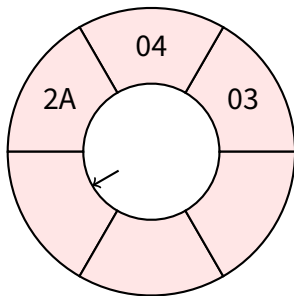


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

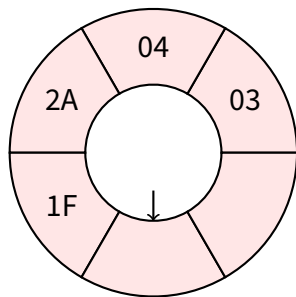


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

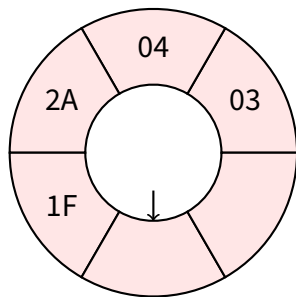


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

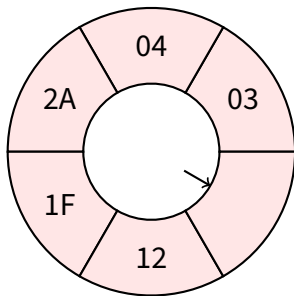


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

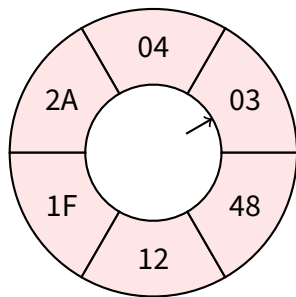


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

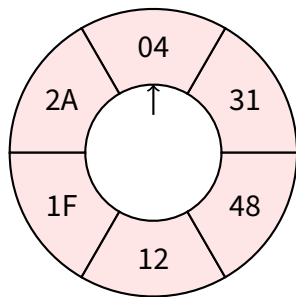


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

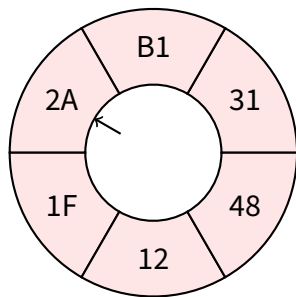


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

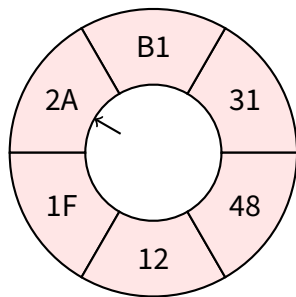


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

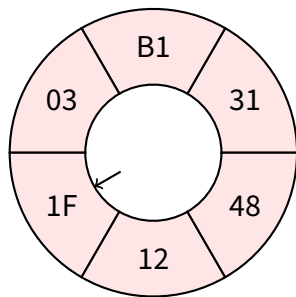


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

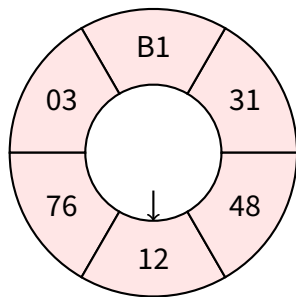


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

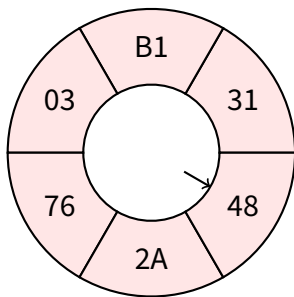


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

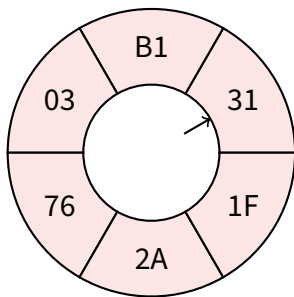


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

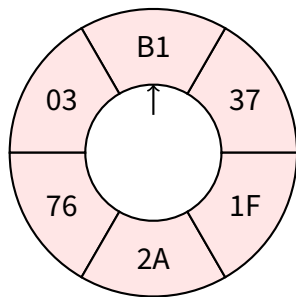


ALGORITHMES TYPE FIFO

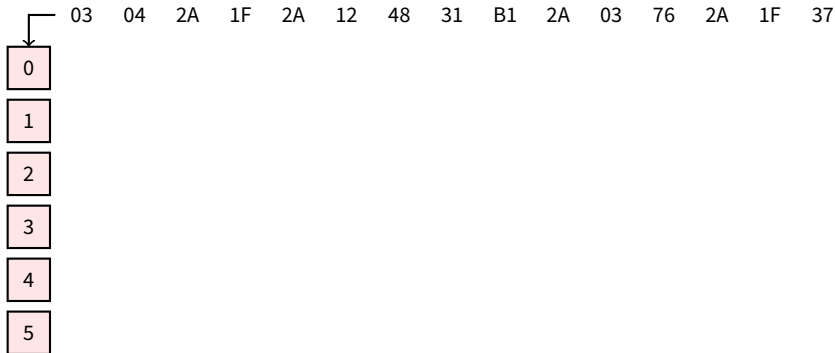
Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

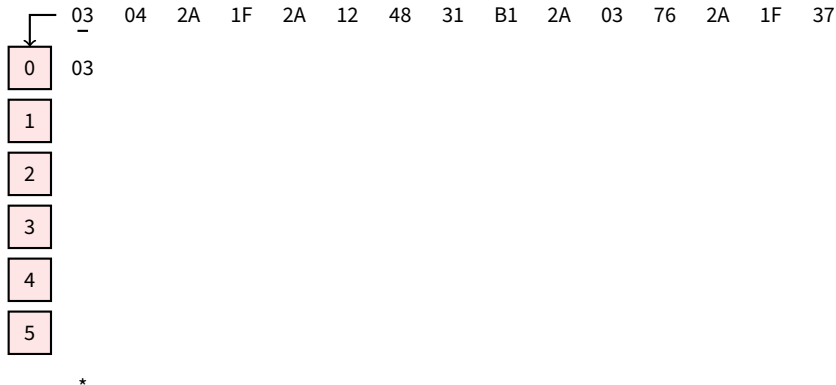
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



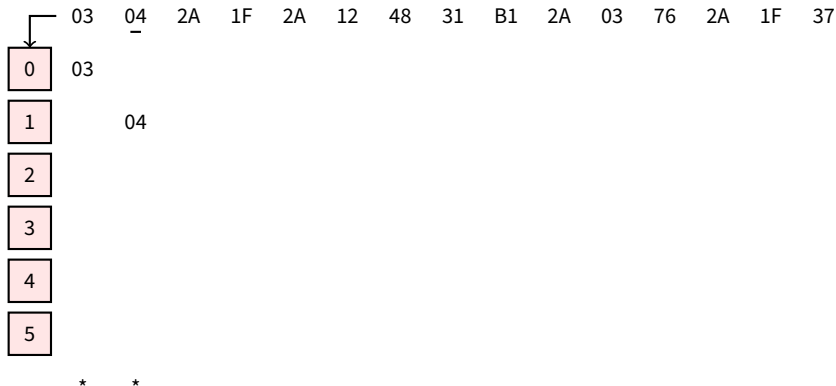
COMPTER LES DÉFAUTS



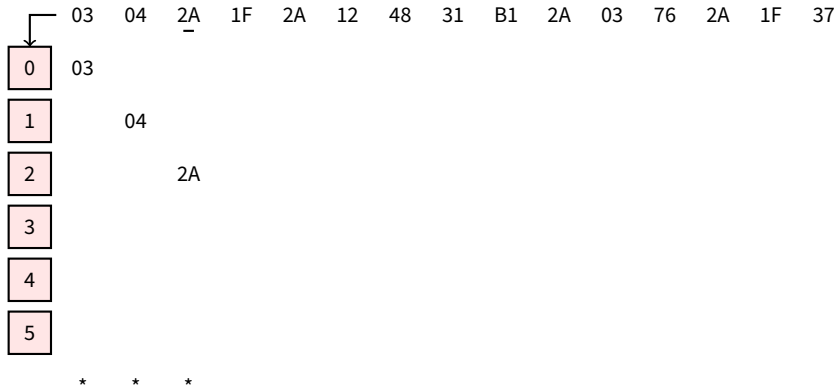
COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS

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

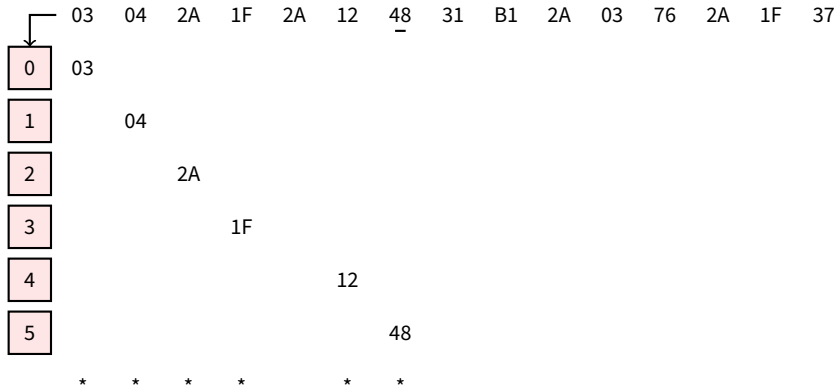
COMPTER LES DÉFAUTS

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

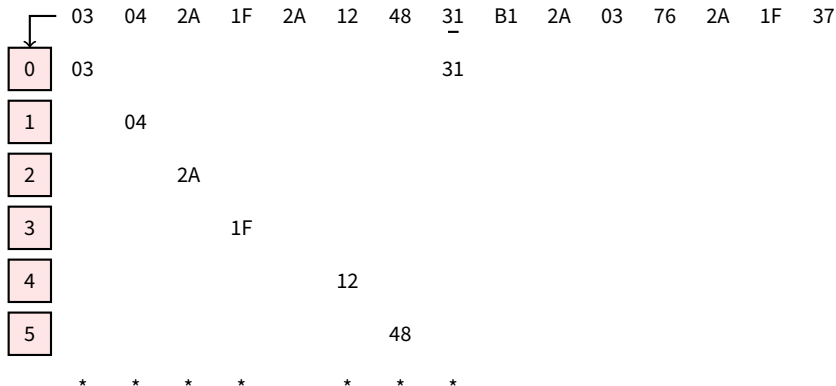
COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

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

COMPTER LES DÉFAUTS

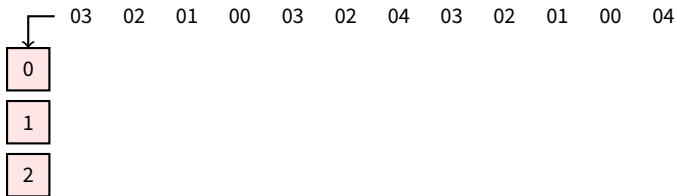
	03	04	2A	1F	2A	12	48	31	B1	2A	03	76	2A	1F	37
0	03							31							37
1		04							B1						
2			2A								03				
3				1F								76			
4						12							2A		
5							48							1F	
	*	*	*	*		*	*	*	*		*	*	*	*	*

Total : **13** défauts

ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres \rightarrow plus de défauts!

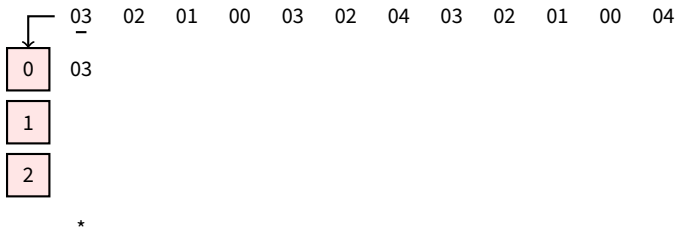
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

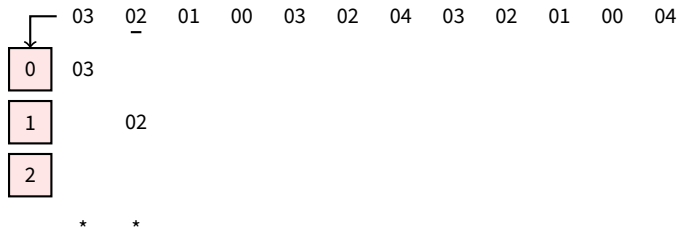
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

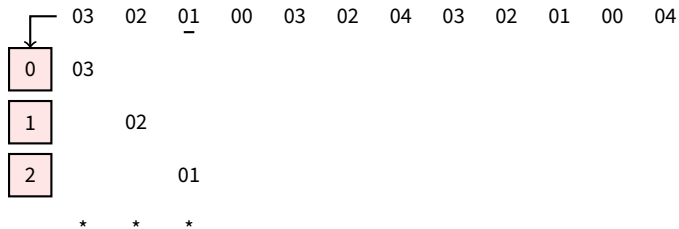
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

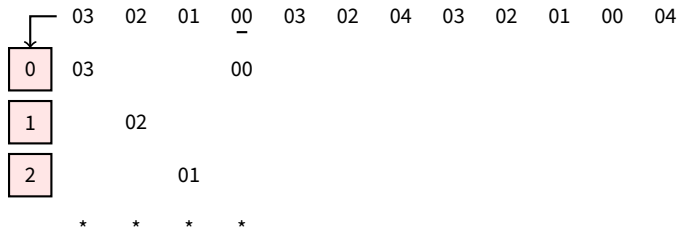
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

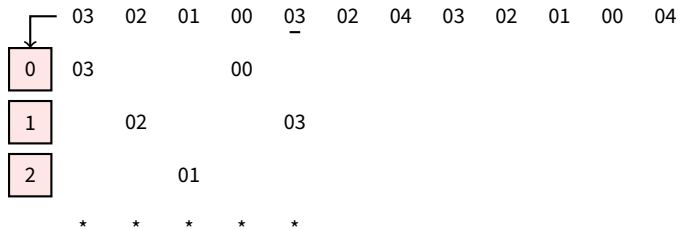
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

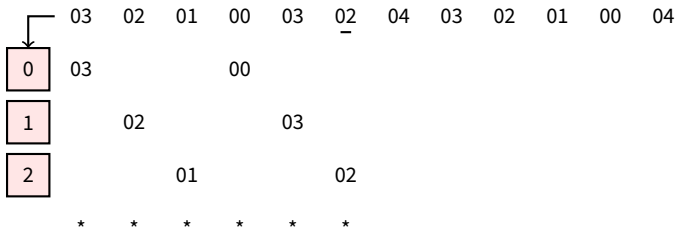
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...

plus de cadres → plus de défauts!

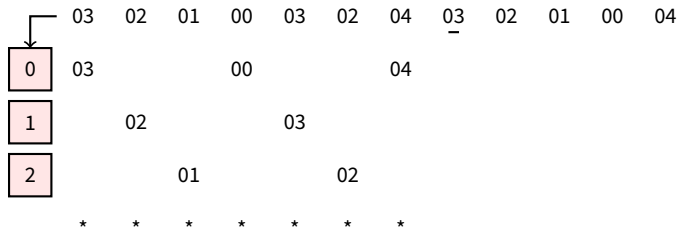
Exemple : avec 3 cadres

	03	02	01	00	03	02	<u>04</u>	03	02	01	00	04
0	03			00			04					
1		02			03							
2			01			02						
	*	*	*	*	*	*	*					

ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

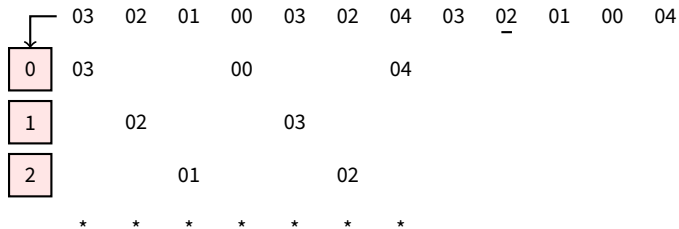
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

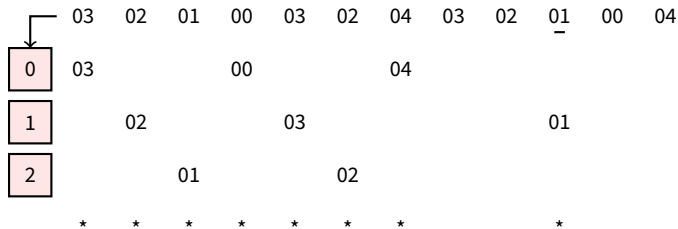
Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 3 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 3 cadres

	03	02	01	00	03	02	04	03	02	01	<u>00</u>	04
0	03			00			04					
1		02			03					01		
2			01			02					00	
	*	*	*	*	*	*	*			*	*	

ANOMALIE DE BELADY

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

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 3 cadres

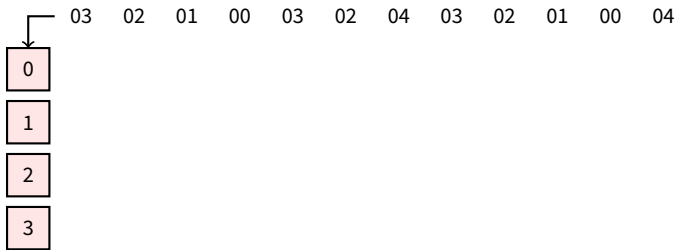
	03	02	01	00	03	02	04	03	02	01	00	04
0	03			00			04					
1		02			03					01		
2			01			02					00	
	*	*	*	*	*	*	*			*	*	

Total : 9 défauts

ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

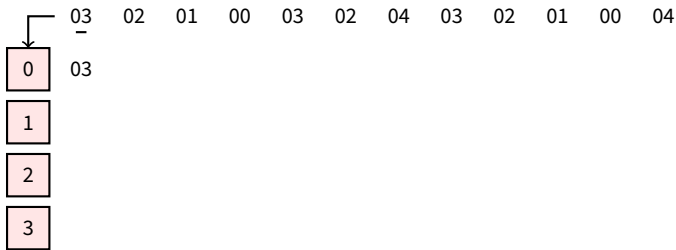
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 4 cadres



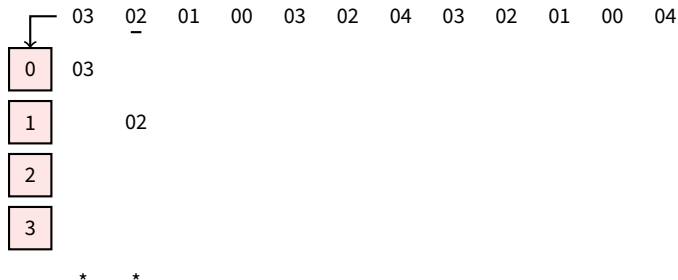
*

ANOMALIE DE BELADY

Sur certaines instances...

plus de cadres → plus de défauts!

Exemple : avec 4 cadres

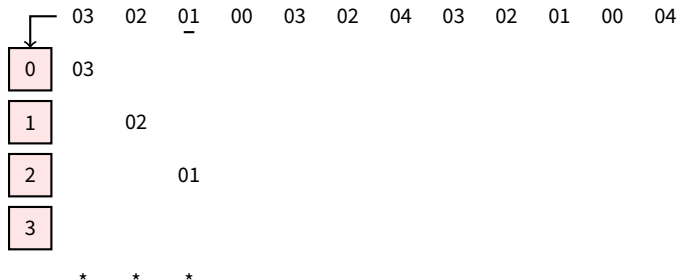


ANOMALIE DE BELADY

Sur certaines instances...

plus de cadres \rightarrow plus de défauts!

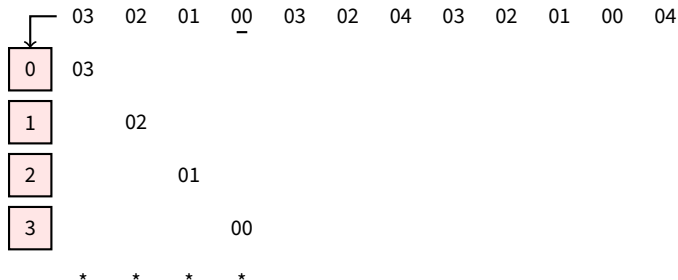
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

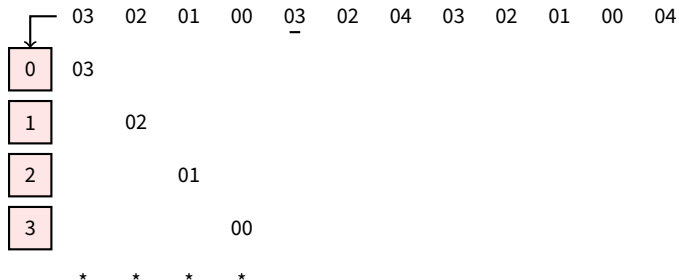
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

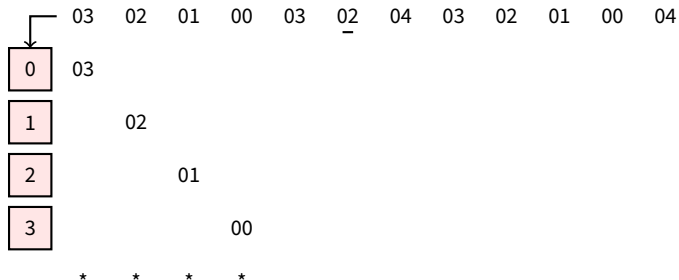
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

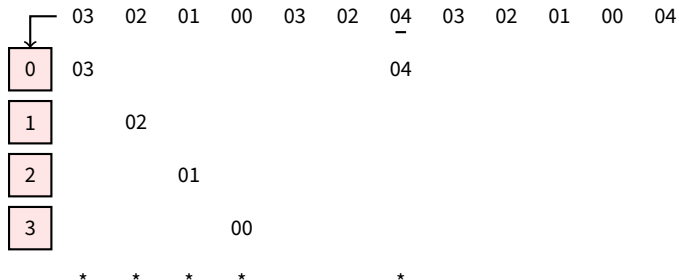
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

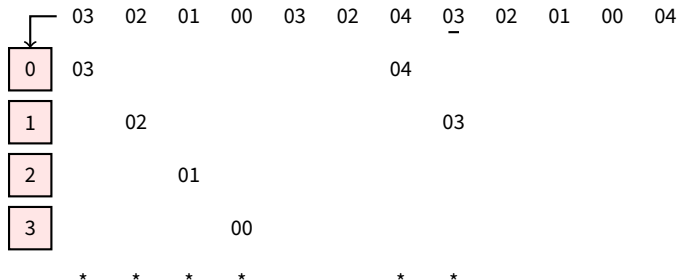
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

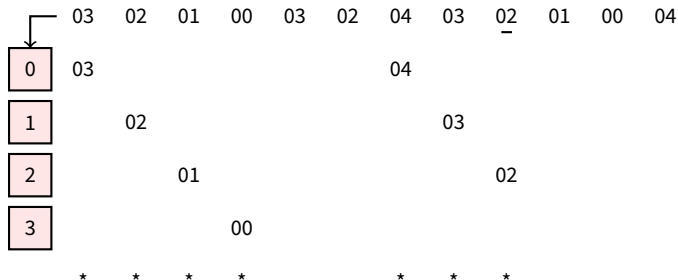
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 4 cadres

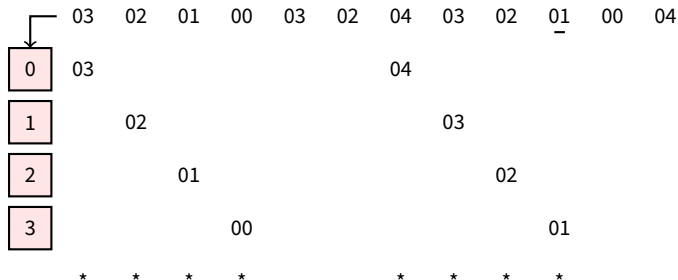


ANOMALIE DE BELADY

Sur certaines instances...

plus de cadres \rightarrow plus de défauts!

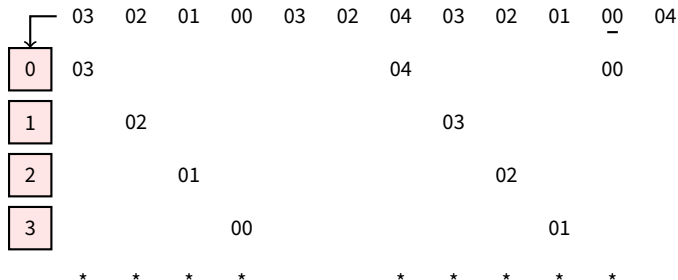
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

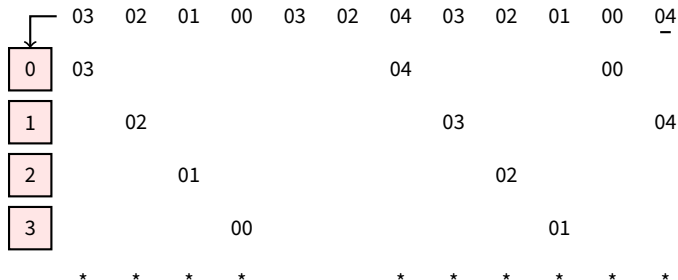
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

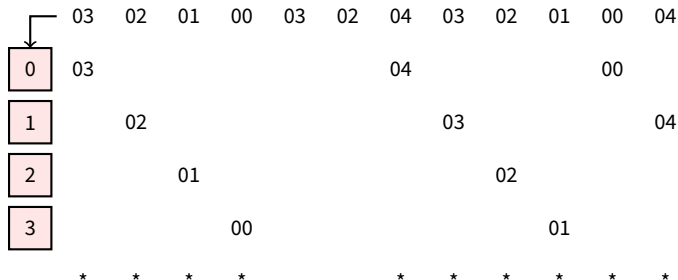
Exemple : avec 4 cadres



ANOMALIE DE BELADY

Sur certaines instances...
plus de cadres → plus de défauts!

Exemple : avec 4 cadres



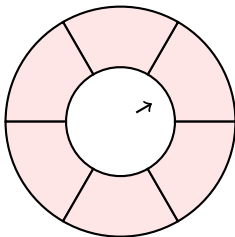
Total : 10 défauts

FIFO - 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

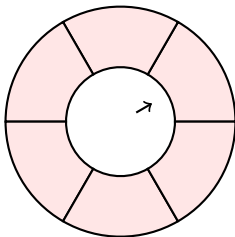


FIFO - SECONDE CHANCE

- 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

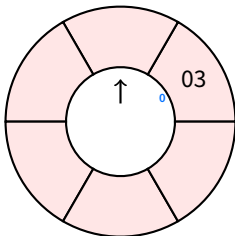


FIFO - SECONDE CHANCE

- 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

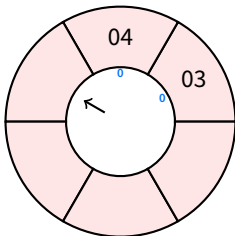


FIFO - SECONDE CHANCE

- 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

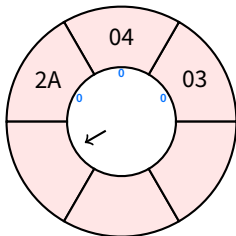


FIFO - SECONDE CHANCE

- 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

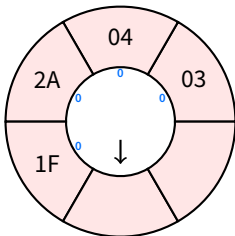


FIFO - SECONDE CHANCE

- 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

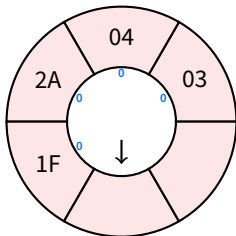


FIFO - SECONDE CHANCE

- 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

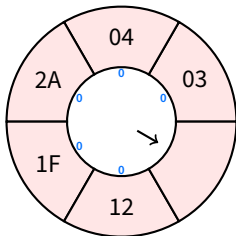


FIFO - SECONDE CHANCE

- 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

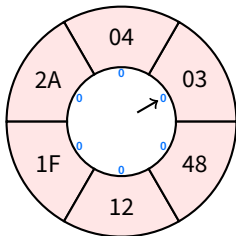


FIFO - SECONDE CHANCE

- 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

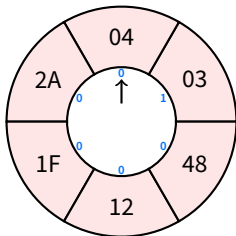


FIFO - SECONDE CHANCE

- 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

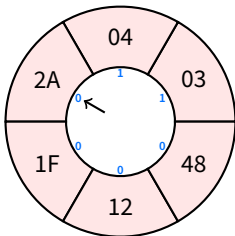


FIFO - SECONDE CHANCE

- 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

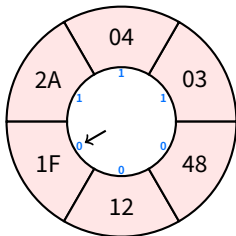


FIFO - SECONDE CHANCE

- 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

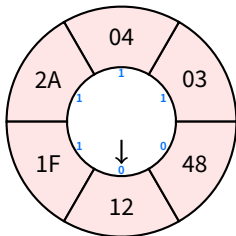


FIFO - SECONDE CHANCE

- 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

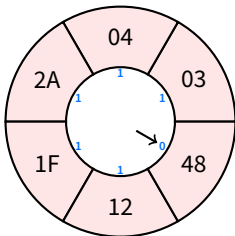


FIFO - SECONDE CHANCE

- 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

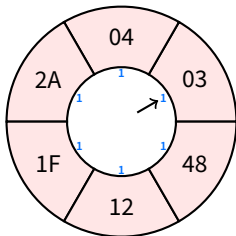


FIFO - SECONDE CHANCE

- 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

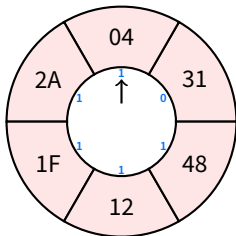


FIFO - SECONDE CHANCE

- 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

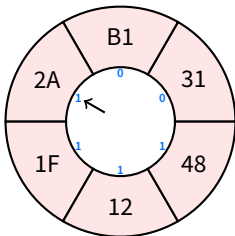


FIFO - SECONDE CHANCE

- 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

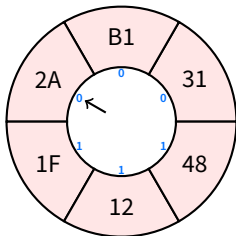


FIFO - SECONDE CHANCE

- 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

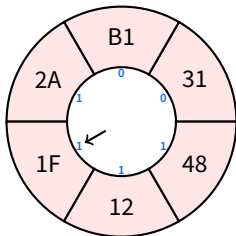


FIFO - SECONDE CHANCE

- 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

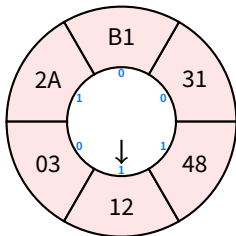


FIFO - SECONDE CHANCE

- 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

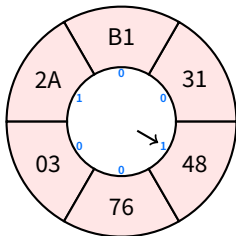


FIFO - SECONDE CHANCE

- 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

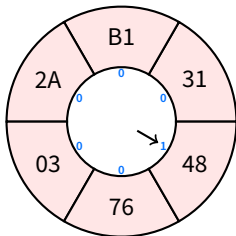


FIFO - SECONDE CHANCE

- 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

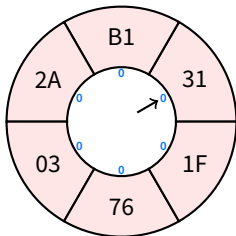


FIFO - SECONDE CHANCE

- 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

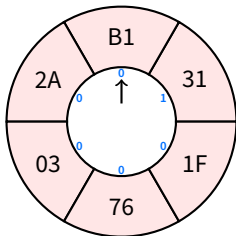


FIFO - SECONDE CHANCE

- 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

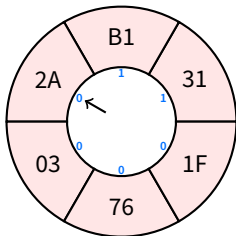


FIFO - SECONDE CHANCE

- 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

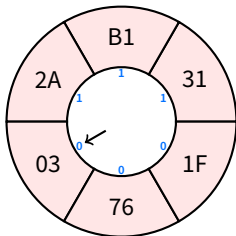


FIFO - SECONDE CHANCE

- 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

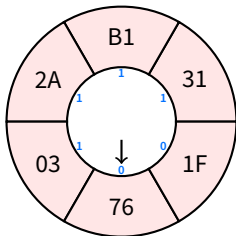


FIFO - SECONDE CHANCE

- 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

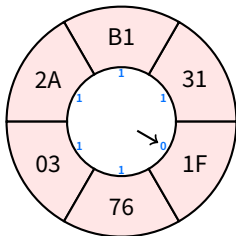


FIFO - SECONDE CHANCE

- 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

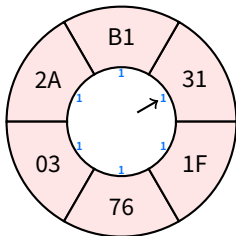


FIFO - SECONDE CHANCE

- 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

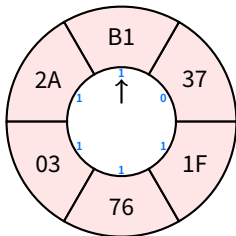


FIFO - SECONDE CHANCE

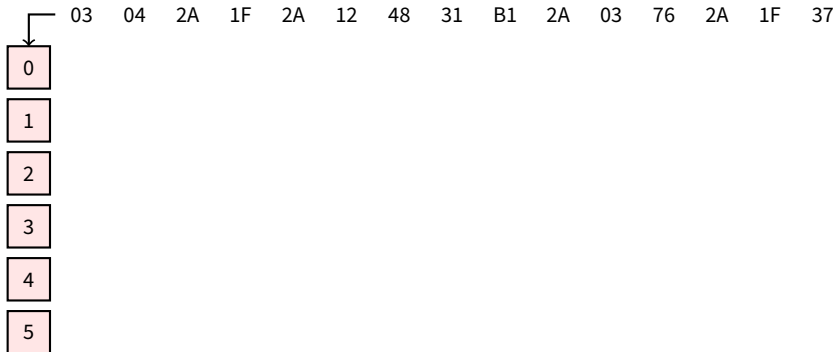
- Principe

- 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)

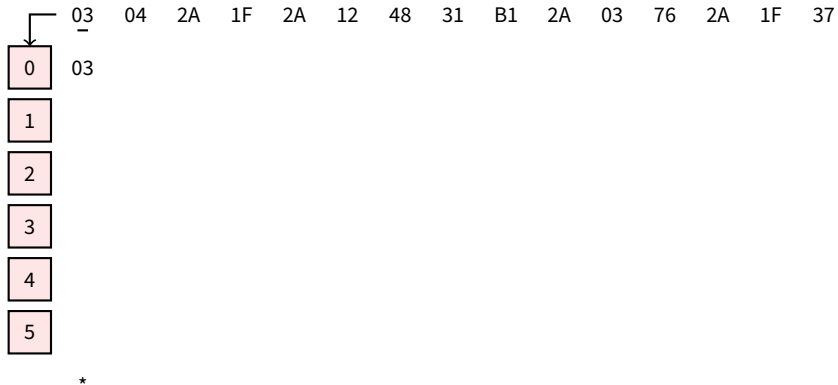
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



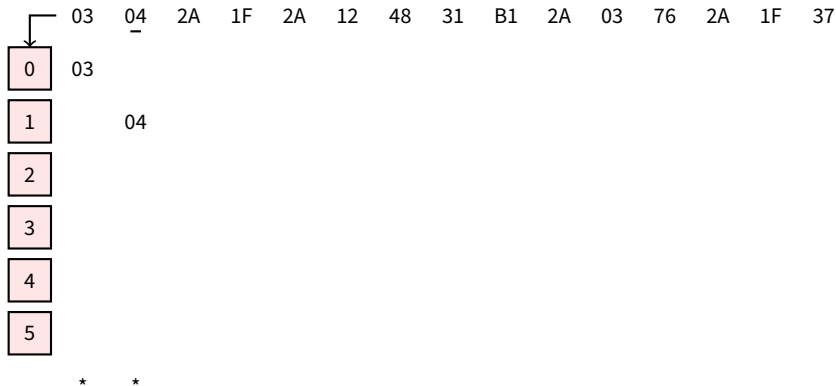
FIFO - SECONDE CHANCE



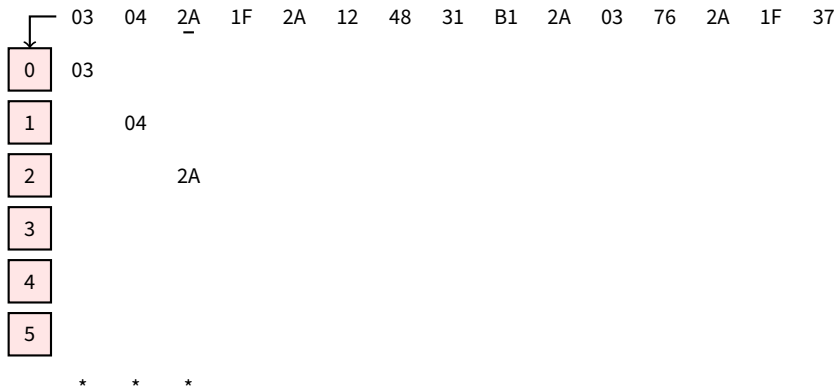
FIFO - SECONDE CHANCE



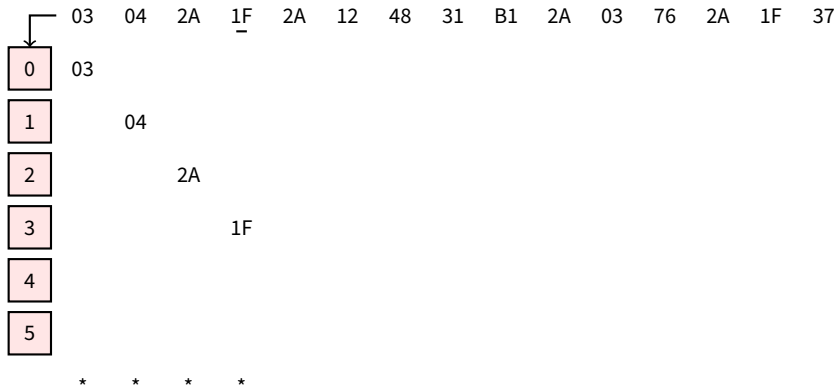
FIFO - SECONDE CHANCE



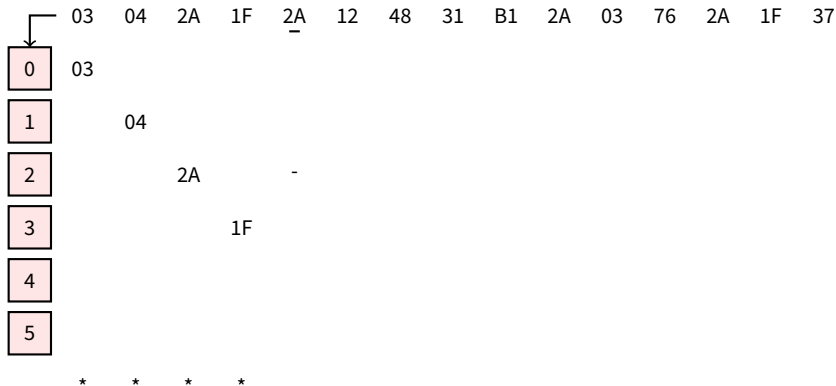
FIFO - SECONDE CHANCE



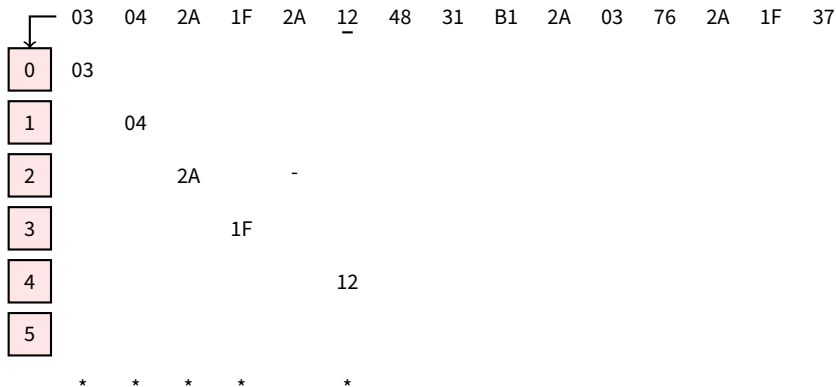
FIFO - SECONDE CHANCE



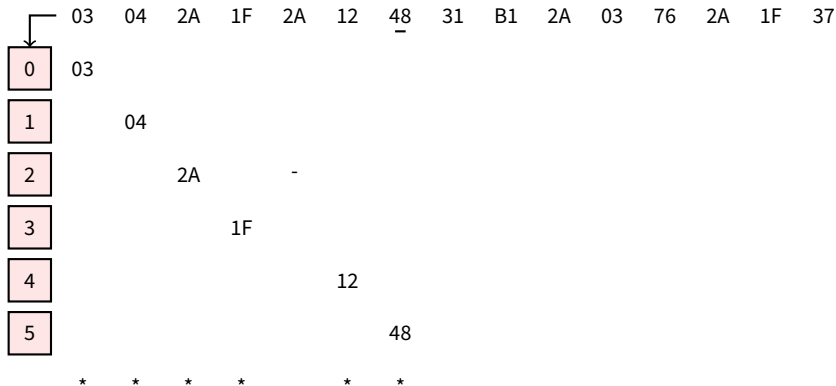
FIFO - SECONDE CHANCE



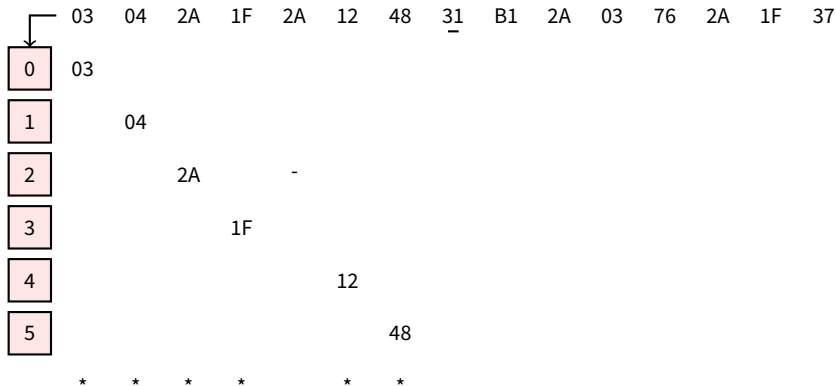
FIFO - SECONDE CHANCE



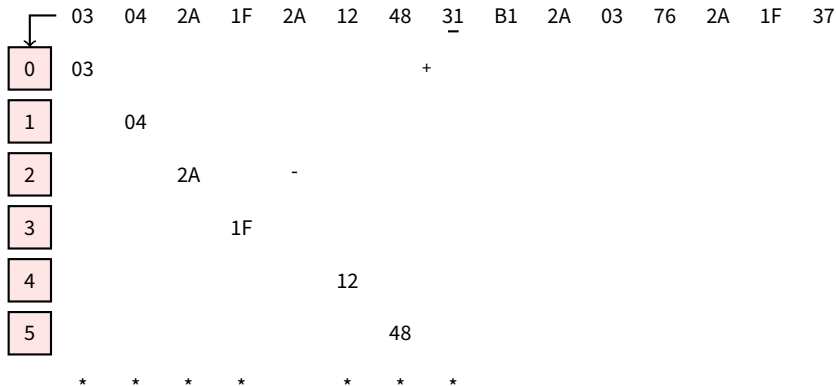
FIFO - SECONDE CHANCE



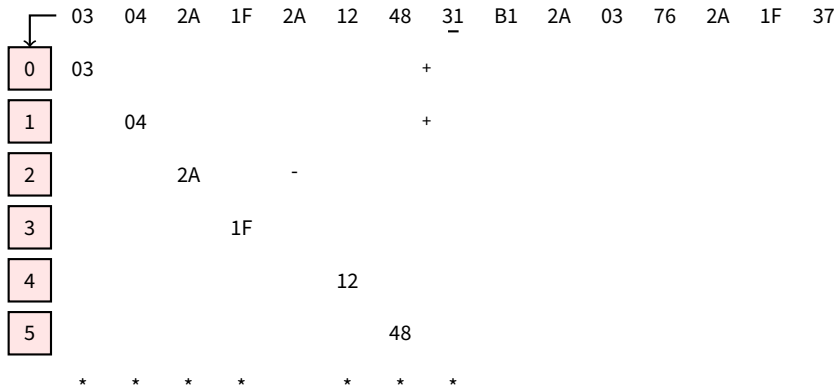
FIFO - SECONDE CHANCE



FIFO - SECONDE CHANCE



FIFO - SECONDE CHANCE



FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

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

FIFO - SECONDE CHANCE

	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	+	
	*	*	*	*		*	*	*	*		*	*		*	*	

LEAST FREQUENTLY USED - LFU

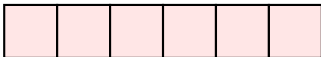
- **Principe**

- 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 - LFU

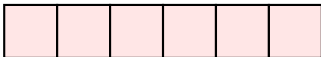
- **Principe**

- 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 - LFU

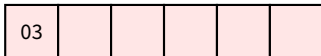
- **Principe**

- 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 - LFU

- **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	

LEAST FREQUENTLY USED - LFU

- 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	

LEAST FREQUENTLY USED - LFU

- 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 - LFU

- 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 - LFU

- 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 - LFU

- 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 - LFU

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




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 - LFU

- 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 - LFU

- 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 - LFU

- Principe

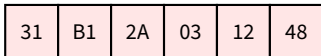
- Noter le taux d'utilisation de chaque page du processus.

-  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 - LFU

- 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 - LFU

- 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 - LFU

- 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 - LFU

- 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

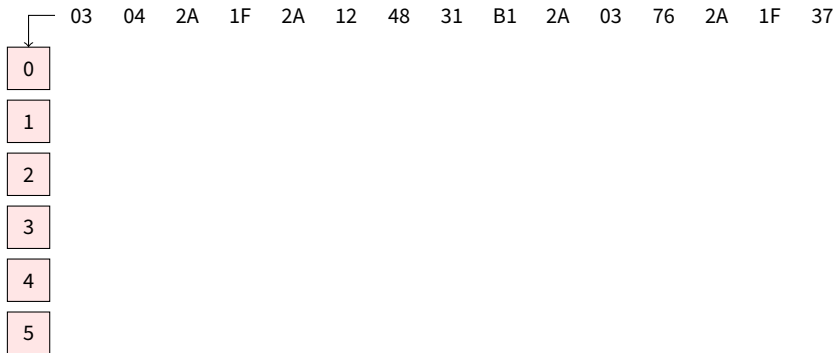
37	B1	2A	03	76	1F
----	----	----	----	----	----



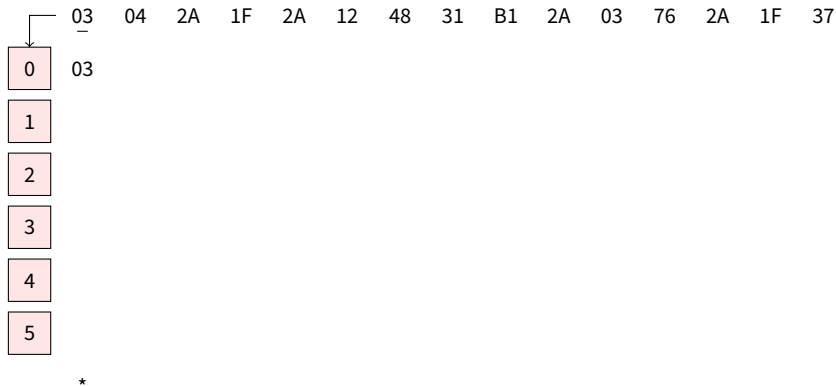
Usage = 1, Date = min({8,9,12})

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

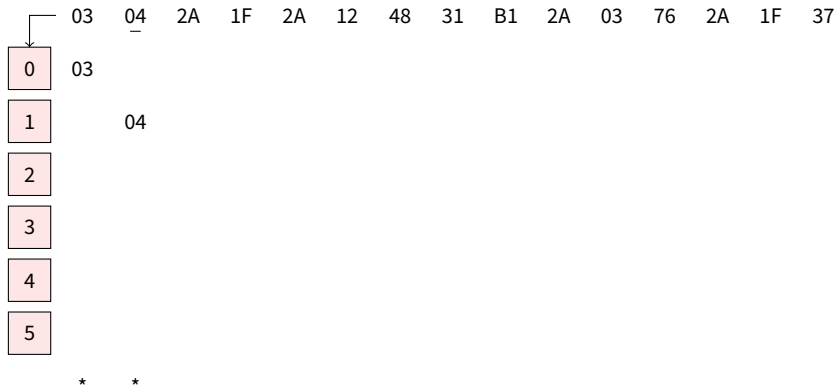
LEAST FREQUENTLY USED - LFU



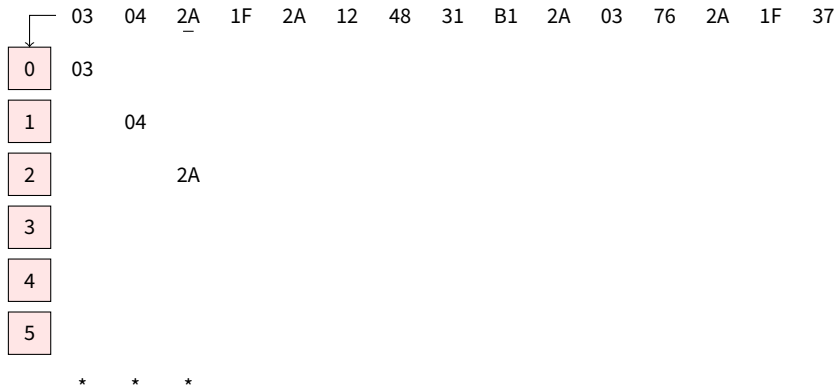
LEAST FREQUENTLY USED - LFU



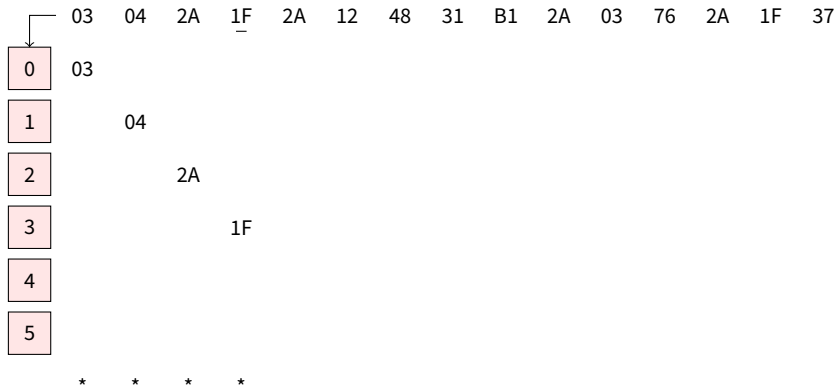
LEAST FREQUENTLY USED - LFU



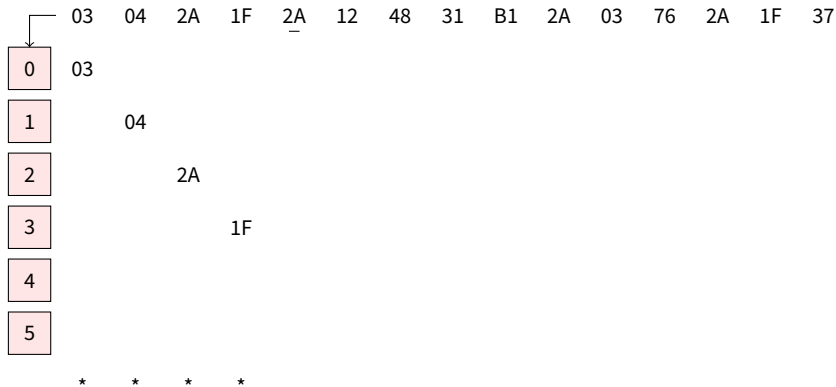
LEAST FREQUENTLY USED - LFU



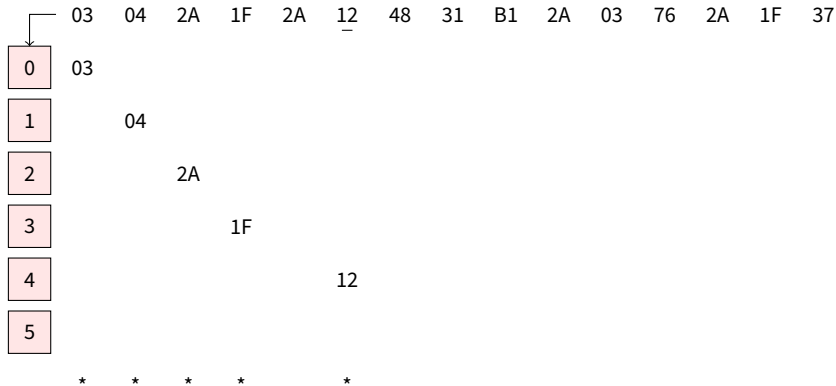
LEAST FREQUENTLY USED - LFU



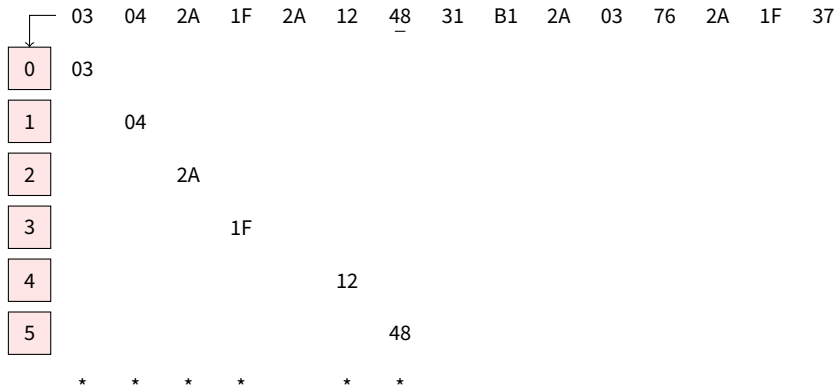
LEAST FREQUENTLY USED - LFU



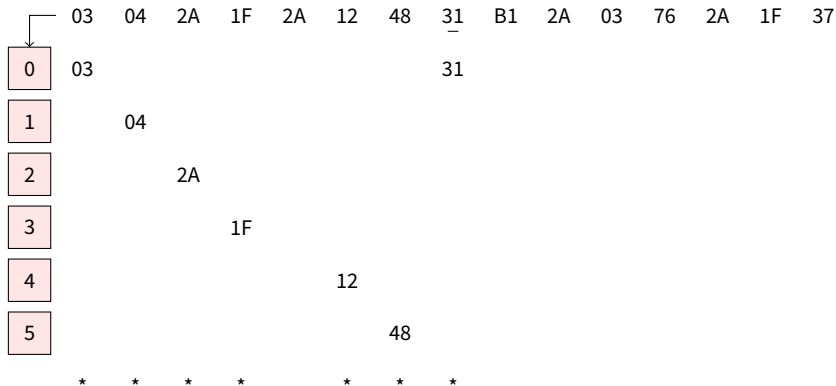
LEAST FREQUENTLY USED - LFU



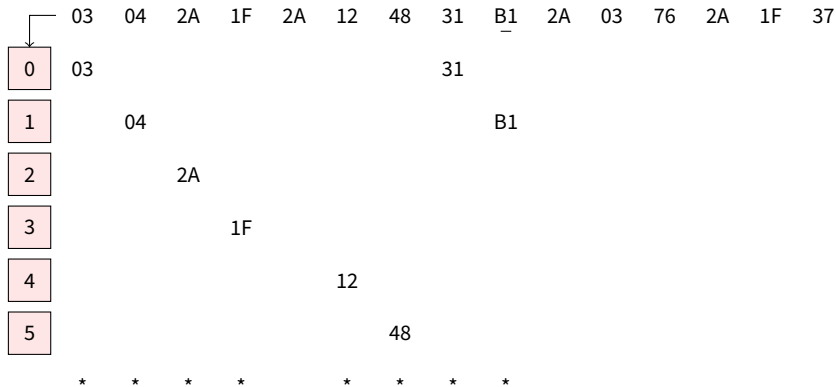
LEAST FREQUENTLY USED - LFU



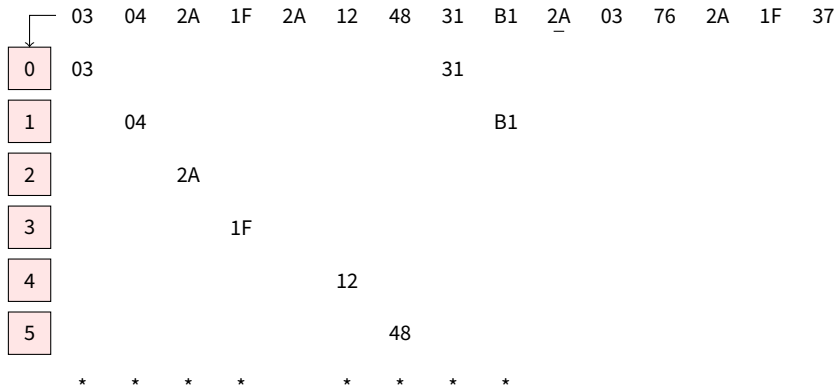
LEAST FREQUENTLY USED - LFU



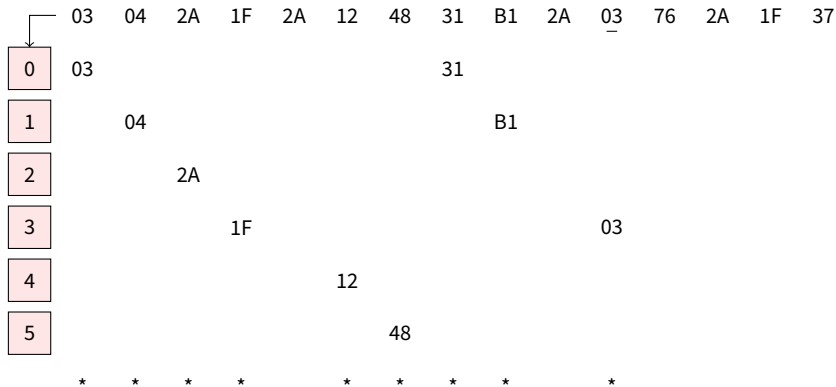
LEAST FREQUENTLY USED - LFU



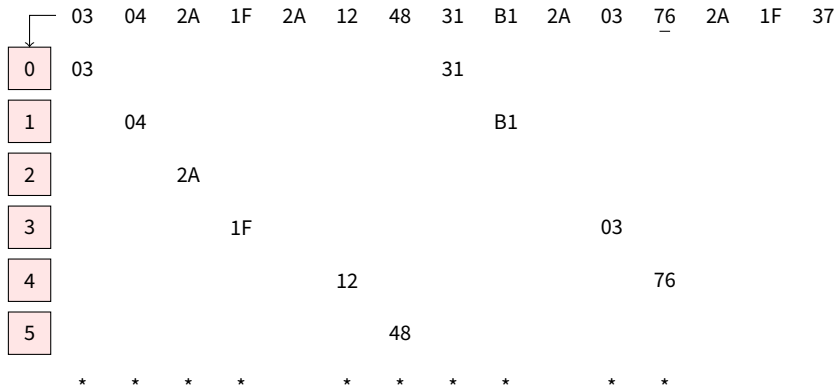
LEAST FREQUENTLY USED - LFU



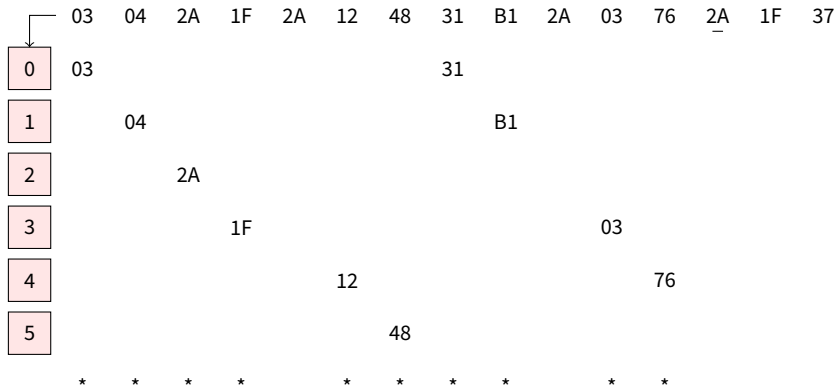
LEAST FREQUENTLY USED - LFU



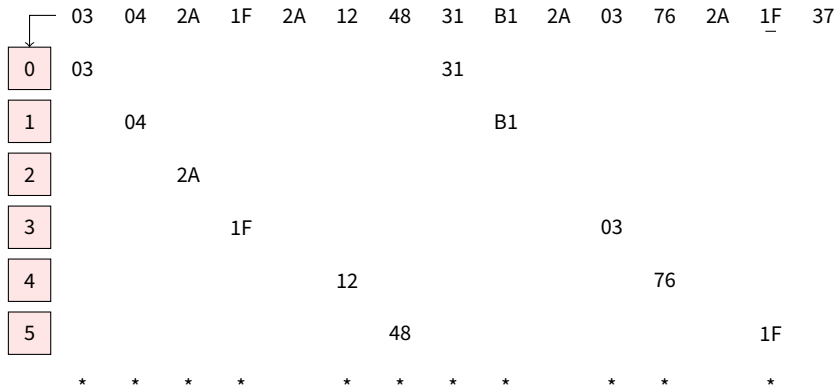
LEAST FREQUENTLY USED - LFU



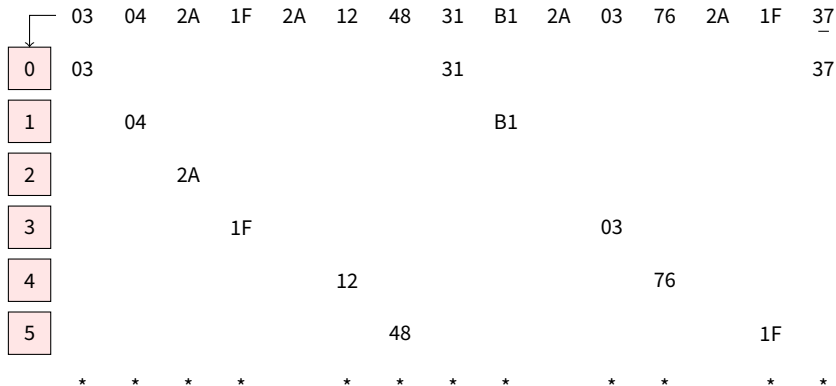
LEAST FREQUENTLY USED - LFU



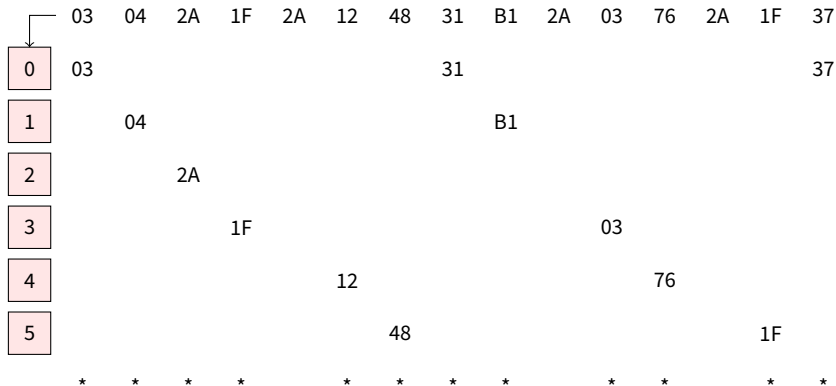
LEAST FREQUENTLY USED - LFU



LEAST FREQUENTLY USED - LFU



LEAST FREQUENTLY USED - LFU



Total : 12 défauts

LEAST FREQUENTLY USED - LFU

LEAST FREQUENTLY USED - LFU

- Nombre total d'utilisation

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

✗ une page beaucoup utilisée reste toujours

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

- **Performance**

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

- **Performance**

- ✓ Beaucoup moins de défaut de page que les autres méthodes

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

- **Performance**

- ✓ Beaucoup moins de défaut de page que les autres méthodes
- ✗ Temps de calcul + mémoire

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

- **Performance**

- ✓ Beaucoup moins de défaut de page que les autres méthodes
- ✗ Temps de calcul + mémoire
 - ✗ 64 bits de plus dans chaque ligne de la table des pages

LEAST FREQUENTLY USED - LFU

- **Nombre total d'utilisation**

- ✗ une page beaucoup utilisée reste toujours
- ✓ remettre usage à 0 périodiquement

- **Performance**

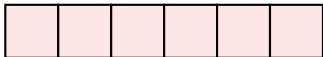
- ✓ Beaucoup moins de défaut de page que les autres méthodes
- ✗ Temps de calcul + mémoire
 - ✗ 64 bits de plus dans chaque ligne de la table des pages
 - ✗ $\mathcal{O}(n)$ opérations à chaque page manquante

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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



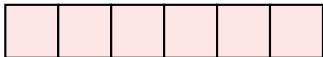
Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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



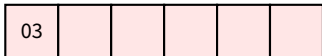
Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **réf**érencée) et M (page **mod**ifiée)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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



R=1
M=0

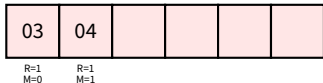
Page	R	M	Date
03	1	0	1
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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			
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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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		
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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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		
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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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		
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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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

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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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	
R=0 M=0	R=0 M=1	R=1 M=1	R=0 M=1	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	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **réf**érencée) et M (page **mod**ifiée)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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

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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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

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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow 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 - NRU

- Principe

- 2 bits : R (page **réf**érencée) et M (page **mod**ifiée)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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

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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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 - NRU

- Principe

- 2 bits : R (page **référéncée**) et M (page **modifiée**)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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=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 - NRU

- Principe

- 2 bits : R (page **réf**érencée) et M (page **mod**ifiée)
- Lecture ou écriture $\rightarrow R=1$; écriture $\rightarrow M=1$
- 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=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 - NRU

- Principe

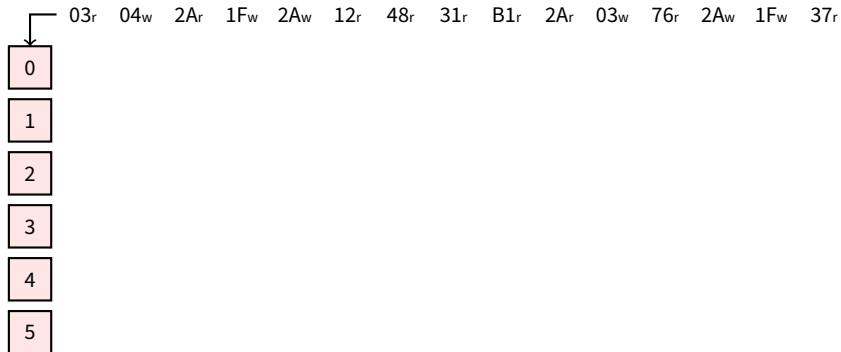
- 2 bits : R (page **réf**érencée) et M (page **mod**ifiée)
- Lecture ou écriture \rightarrow R=1; écriture \rightarrow M=1
- Priorité : $(R=1, M=1) > (R=1, M=0) > (R=0, M=1) > (R=0, M=0)$
- FIFO en cas d'égalité
- Tous les R sont remis à 0 **chaque K cycles**

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

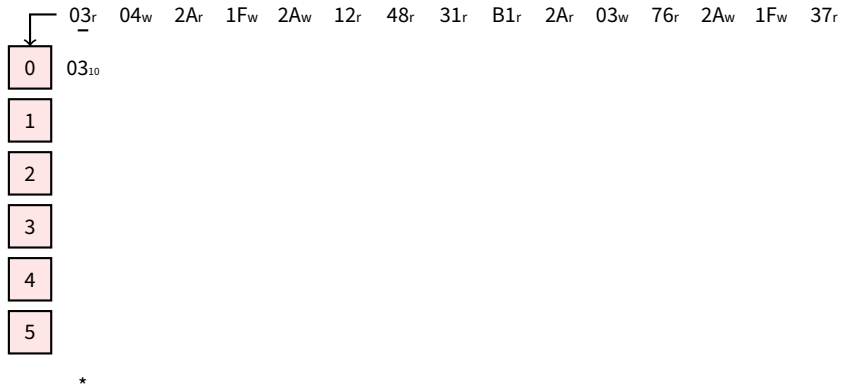
76	04	2A	1F	37	03
R=0 M=0	R=0 M=1	R=1 M=1	R=1 M=1	R=1 M=0	R=0 M=1

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

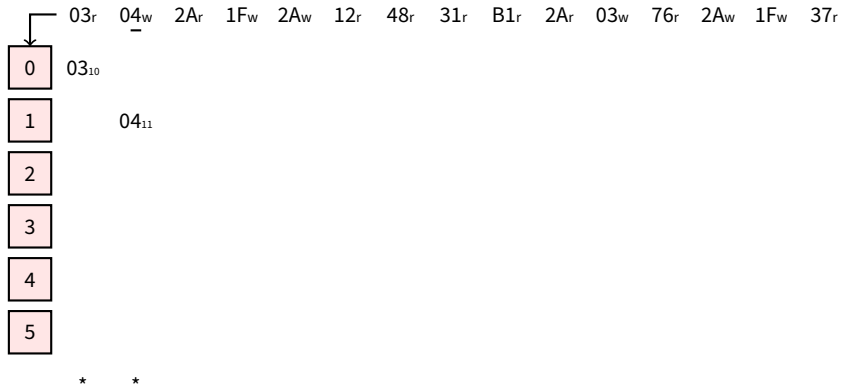
NOT RECENTLY USED - NRU



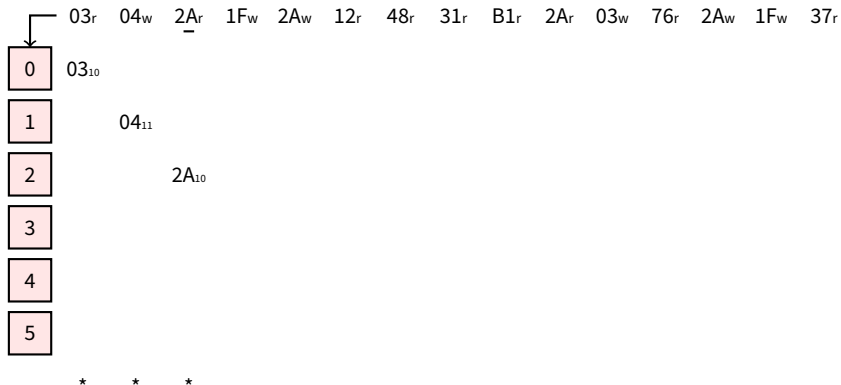
NOT RECENTLY USED - NRU



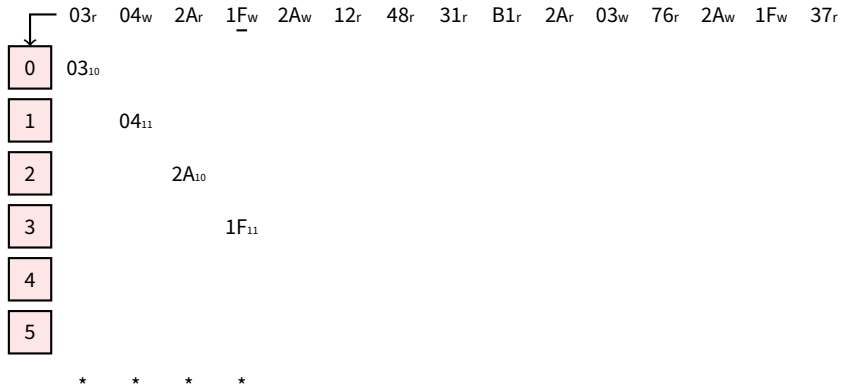
NOT RECENTLY USED - NRU



NOT RECENTLY USED - NRU



NOT RECENTLY USED - NRU



NOT RECENTLY USED - NRU

	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
0	03 ₁₀				00										
1		04 ₁₁			01										
2			2A ₁₀		00										
3				1F ₁₁	01										
4															
5															
	*	*	*	*											

NOT RECENTLY USED - NRU

	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
0	03 ₁₀				00										
1		04 ₁₁			01										
2			2A ₁₀		00	11									
3				1F ₁₁	01										
4															
5															
	*	*	*	*											

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	<u>12</u> _r	48 _r	31 _r	B1 _r	2A _r	03 _w	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀				00										
1		04 ₁₁			01										
2			2A ₁₀		00	11									
3				1F ₁₁	01										
4						12 ₁₀									
5															
	*	*	*	*		*									

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	<u>48_r</u>	31 _r	B1 _r	2A _r	03 _w	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀				00										
1		04 ₁₁			01										
2			2A ₁₀		00	11									
3				1F ₁₁	01										
4						12 ₁₀									
5							48 ₁₀								
	*	*	*	*		*	*								

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	<u>31</u> _r	B1 _r	2A _r	03 _w	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀							
1		04 ₁₁		01											
2			2A ₁₀	00	11										
3				1F ₁₁ 01											
4						12 ₁₀									
5							48 ₁₀								
	*	*	*	*		*	*	*							

NOT RECENTLY USED - NRU

	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
0	03 ₁₀			00				31 ₁₀ 00							
1		04 ₁₁		01											
2			2A ₁₀	00	11					01					
3				1F ₁₁ 01											
4						12 ₁₀				00					
5							48 ₁₀			00					
	*	*	*	*		*	*	*							

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	31 _r	<u>B1_r</u>	2A _r	03 _w	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀ 00							
1		04 ₁₁		01											
2			2A ₁₀	00	11				01						
3				1F ₁₁ 01											
4						12 ₁₀			00 B1 ₁₀						
5							48 ₁₀		00						
	*	*	*	*		*	*	*	*						

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	31 _r	B1 _r	<u>2A_r</u>	03 _w	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀ 00							
1		04 ₁₁		01											
2			2A ₁₀	00	11				01		11				
3				1F ₁₁ 01											
4						12 ₁₀			00 B1 ₁₀						
5							48 ₁₀		00						
	*	*	*	*		*	*	*	*						

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	31 _r	B1 _r	2A _r	<u>03_w</u>	76 _r	2A _w	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀ 00							
1		04 ₁₁		01											
2			2A ₁₀	00	11				01		11				
3				1F ₁₁ 01											
4						12 ₁₀			00 B1 ₁₀						
5							48 ₁₀	00			03 ₁₁				
	*	*	*	*		*	*	*	*		*				

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	31 _r	B1 _r	2A _r	03 _w	<u>76_r</u>	2A _w	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11					
3				1F ₁₁ 01											
4						12 ₁₀		00 B1 ₁₀							
5							48 ₁₀	00			03 ₁₁				
	*	*	*	*		*	*	*	*		*	*			

NOT RECENTLY USED - NRU

	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
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀ 00			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11			01		
3				1F ₁₁ 01											
4						12 ₁₀		00 B1 ₁₀					00		
5							48 ₁₀	00			03 ₁₁		01		
	*	*	*	*		*	*	*	*		*	*			

NOT RECENTLY USED - NRU

	03 _r	04 _w	2A _r	1F _w	2A _w	12 _r	48 _r	31 _r	B1 _r	2A _r	03 _w	76 _r	<u>2A_w</u>	1F _w	37 _r
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀ 00			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11			01	11	
3				1F ₁₁ 01											
4						12 ₁₀		00 B1 ₁₀					00		
5							48 ₁₀	00			03 ₁₁		01		
	*	*	*	*		*	*	*	*		*	*			

NOT RECENTLY USED - NRU

	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
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀ 00			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11			01	11	
3				1F ₁₁ 01											11
4						12 ₁₀		00	B1 ₁₀				00		
5							48 ₁₀	00			03 ₁₁		01		
	*	*	*	*		*	*	*	*		*	*			

NOT RECENTLY USED - NRU

	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
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀ 00			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11			01	11	
3				1F ₁₁ 01											11
4						12 ₁₀		00	B1 ₁₀				00		37 ₁₀
5							48 ₁₀	00			03 ₁₁		01		
	★	★	★	★		★	★	★	★		★	★			★

NOT RECENTLY USED - NRU

	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
0	03 ₁₀			00				31 ₁₀ 00				76 ₁₀ 00			
1		04 ₁₁		01											
2			2A ₁₀	00	11			01		11			01	11	
3				1F ₁₁ 01											11
4						12 ₁₀		00	B1 ₁₀				00		37 ₁₀
5							48 ₁₀	00			03 ₁₁		01		
	*	*	*	*		*	*	*	*		*	*			*

Total : 11 défauts

NOT RECENTLY USED - NRU

NOT RECENTLY USED - NRU

- Performance

NOT RECENTLY USED - NRU

- **Performance**

- ✓ Peu de défaut de page

NOT RECENTLY USED - NRU

- **Performance**

- ✓ Peu de défaut de page
- ✓ Peu coûteux en mémoire

NOT RECENTLY USED - NRU

- Performance

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

NOT RECENTLY USED - NRU

- **Performance**

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

- **Gain**

NOT RECENTLY USED - NRU

- **Performance**

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

- **Gain**

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

LEAST RECENTLY USED - LRU

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

LEAST RECENTLY USED - LRU

- 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						

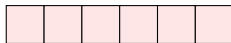
LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
 → 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	0	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	0	
2				0	0	0	
3					0	0	
4						0	
5							



LEAST RECENTLY USED - LRU

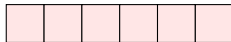
- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
 → remettre la ligne i à 1 puis la colonne i à 0

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

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



LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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					
----	--	--	--	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	0	
2				0	0	0	
3					0	0	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	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				
----	----	--	--	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			1	1	1	1	
2				0	0	0	
3					0	0	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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			
----	----	----	--	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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			
----	----	----	--	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension $N = \text{nombre de cadres}$
- Utilisation d'une page dans le cadre i
 → 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		
----	----	----	----	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	1	1	
2				0	1	1	
3					1	1	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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
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		
----	----	----	----	--	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	1	1	
2				1	1	1	
3					1	1	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	
----	----	----	----	----	--

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	1	
2				1	0	1	
3					0	1	
4						1	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	0
1	1	0	0	0	0	0
2	1	1	0	0	0	0
3	1	1	1	0	0	0
4	1	1	1	1	0	0
5	1	1	1	1	1	0

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

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	0	
2				1	0	0	
3					0	0	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			1	1	1	1	
2				1	0	0	
3					0	0	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension $N = \text{nombre de cadres}$
- Utilisation d'une page dans le cadre i
 → 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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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		03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1		0	1	1	1		
2			1	1	1		
3				0	0		
4					0		
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	1	1	
2				0	1	1	
3					1	1	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	1	
2				0	0	1	
3					0	1	
4						1	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
 → 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
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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	1	
2				1	1	1	
3					0	1	
4						1	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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
----	----	----	----	----	----

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
 → 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	0	03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37
1			0	0	0	0	
2				1	1	0	
3					0	0	
4						0	
5							

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ 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	0
1	1	0	0	0	0	0
2	1	1	0	0	0	0
3	1	1	1	0	0	0
4	1	1	1	1	0	0
5	1	1	1	1	1	0

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

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

LEAST RECENTLY USED - LRU

- Matrice triangulaire de dimension N = nombre de *cadres*
- Utilisation d'une page dans le cadre i
→ remettre la ligne i à 1 puis la colonne i à 0

👉 Cadre le plus ancien (à utiliser) =

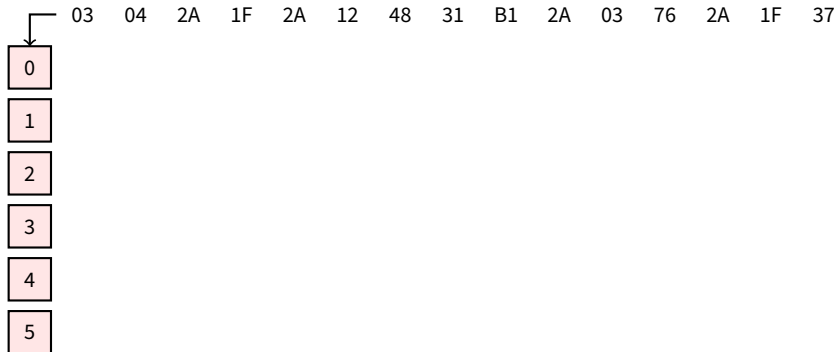
ligne est remplie de 0, colonne remplie de 1 (sauf ligne i)

	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						

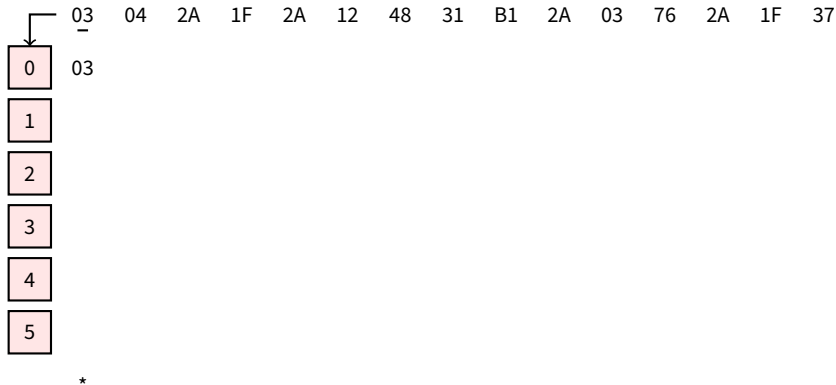
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37

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

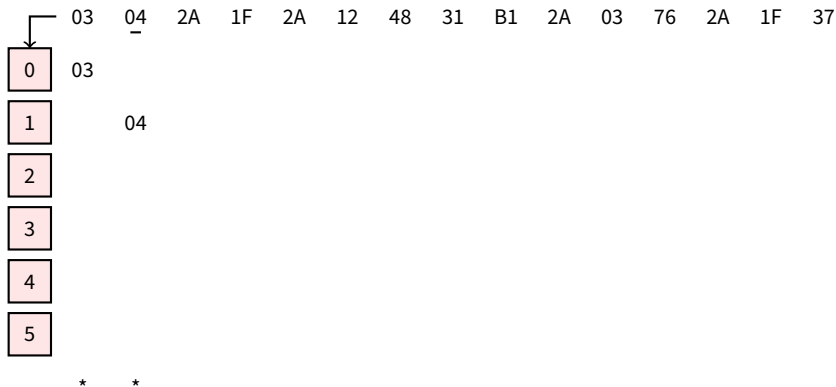
LEAST RECENTLY USED - LRU



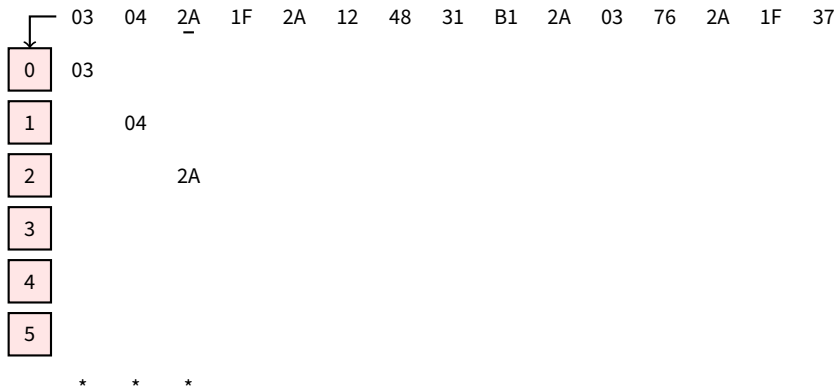
LEAST RECENTLY USED - LRU



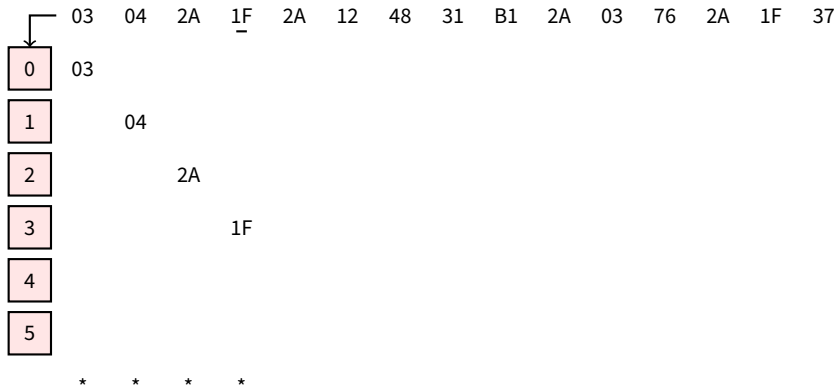
LEAST RECENTLY USED - LRU



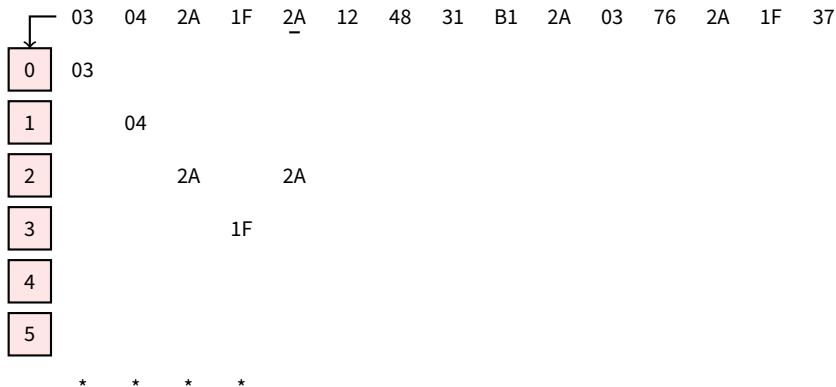
LEAST RECENTLY USED - LRU



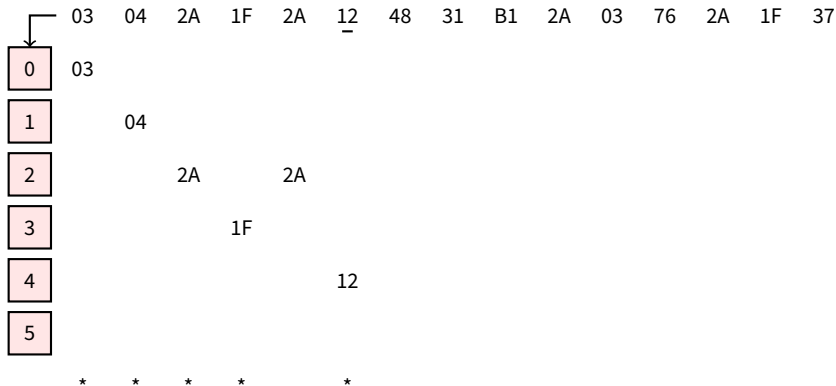
LEAST RECENTLY USED - LRU



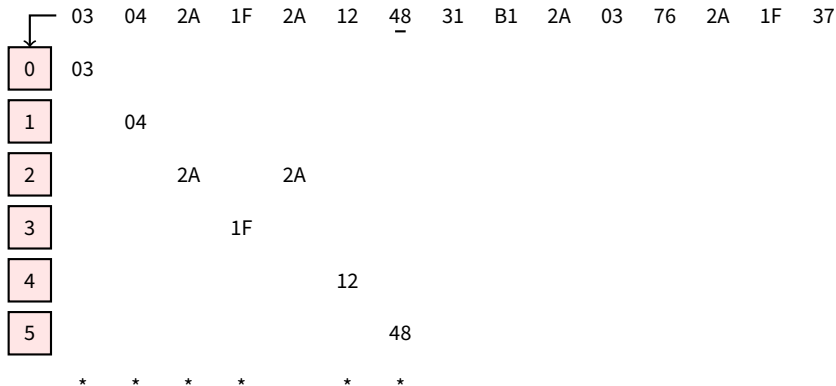
LEAST RECENTLY USED - LRU



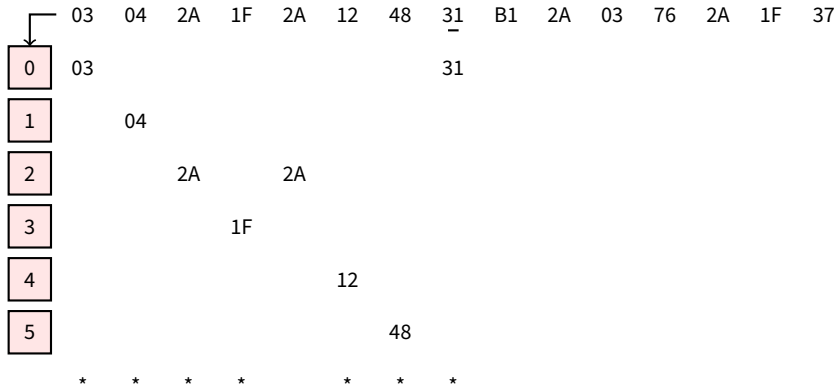
LEAST RECENTLY USED - LRU



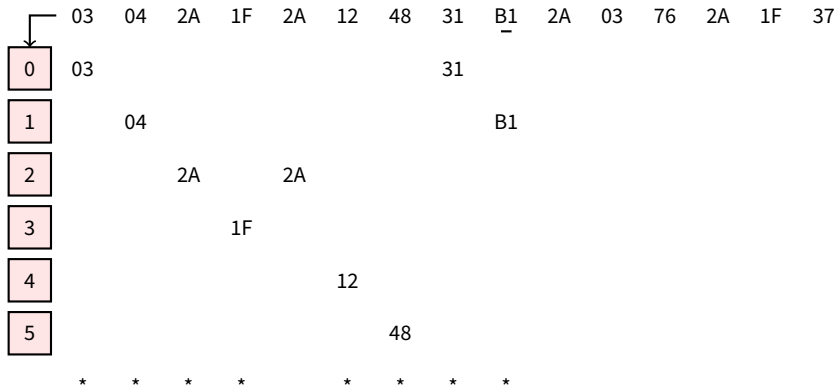
LEAST RECENTLY USED - LRU



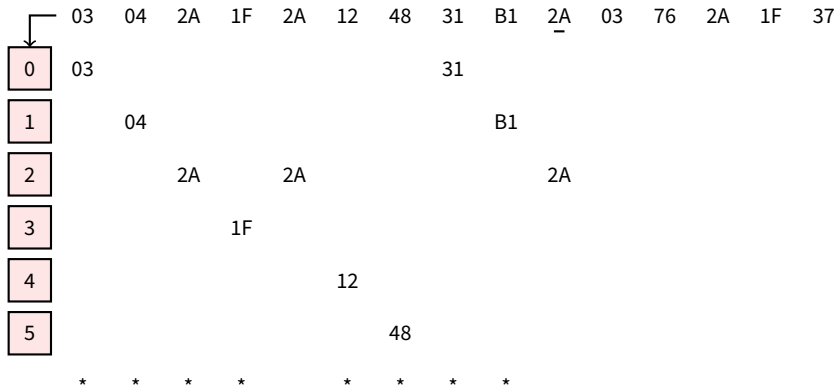
LEAST RECENTLY USED - LRU



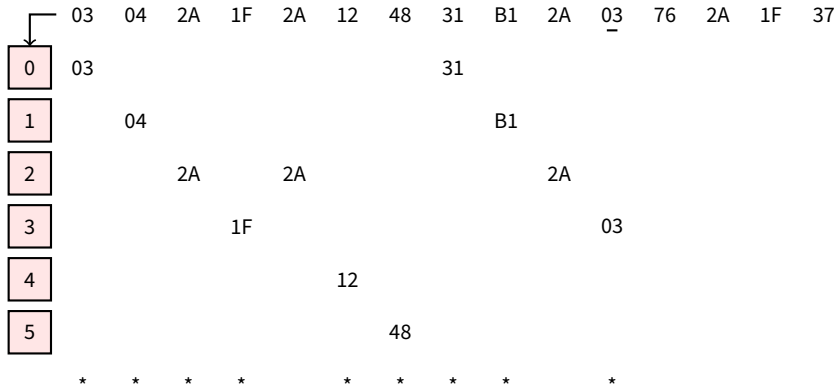
LEAST RECENTLY USED - LRU



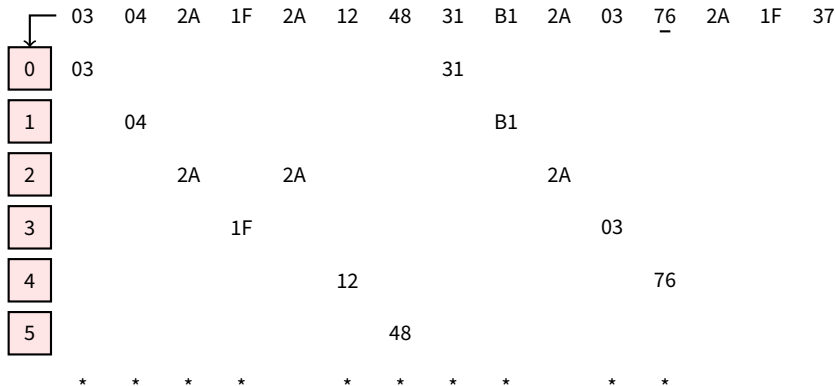
LEAST RECENTLY USED - LRU



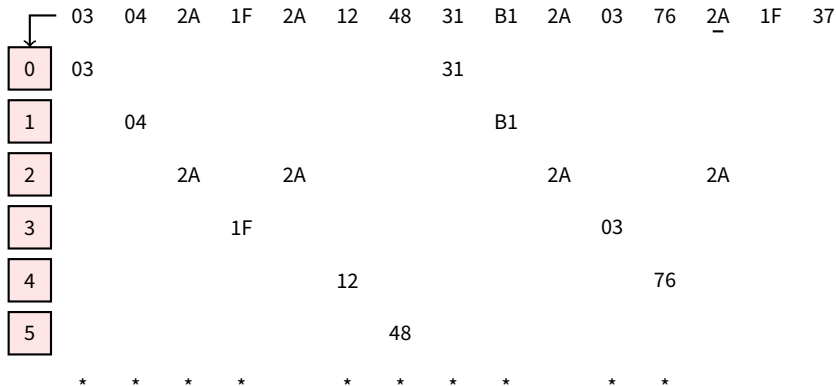
LEAST RECENTLY USED - LRU



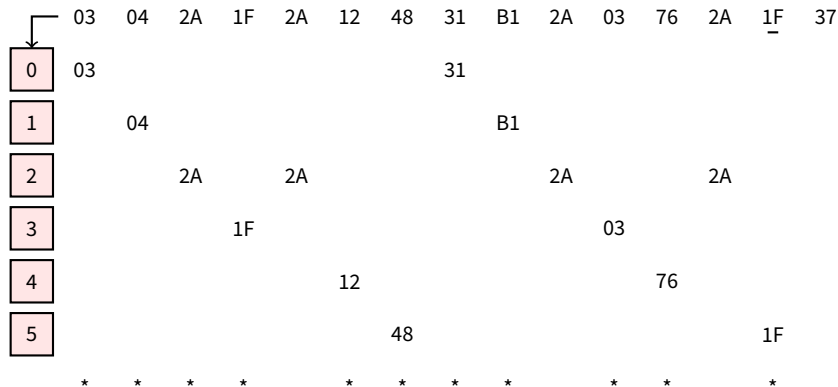
LEAST RECENTLY USED - LRU



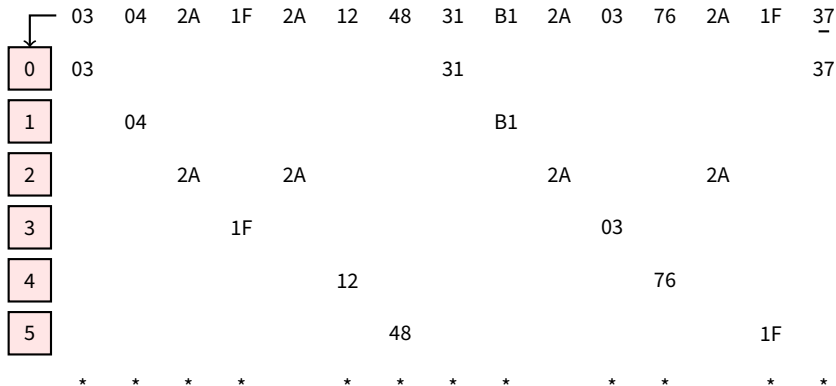
LEAST RECENTLY USED - LRU



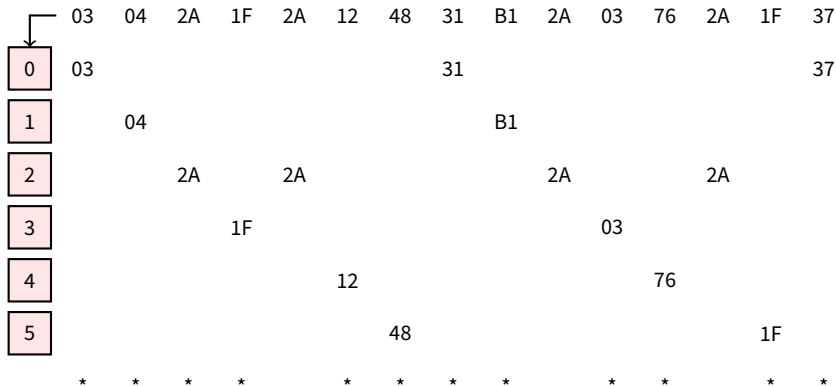
LEAST RECENTLY USED - LRU



LEAST RECENTLY USED - LRU



LEAST RECENTLY USED - LRU



LIMITES DE LA MÉMOIRE PAGINÉE

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle
- Chargement d'une page

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle
- Chargement d'une page
 - Plein de données inutiles

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle
- Chargement d'une page
 - Plein de données inutiles
 - Pas forcément tout ce dont on a besoin

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle
- Chargement d'une page
 - Plein de données inutiles
 - Pas forcément tout ce dont on a besoin
- Idée → **mémoire segmentée**

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus
 - Taille arbitraire ($2^{nb_bits_cadre}$)
 - 👉 peut couper une portion de code, un bloc de données ...
 - Fragmentation résiduelle
- Chargement d'une page
 - Plein de données inutiles
 - Pas forcément tout ce dont on a besoin
- Idée → **mémoire segmentée**
 - Découper en tenant compte de la structure du processus (code + données)

PLAN

- La mémoire
- La mémoire virtuelle
- La mémoire segmentée
- Synthèse

[Retour à l'accueil](#) - [Retour au plan](#)

MÉMOIRE SEGMENTÉE

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

- Pile

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

- Pile
- Table des symboles

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

- Pile
- Table des symboles
- Programme principal

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

- Pile
- Table des symboles
- Programme principal
- Fonction

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

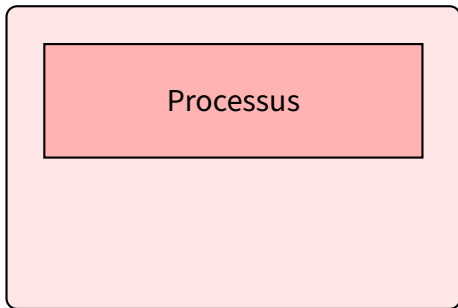
- Pile
- Table des symboles
- Programme principal
- Fonction
- Bibliothèque

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

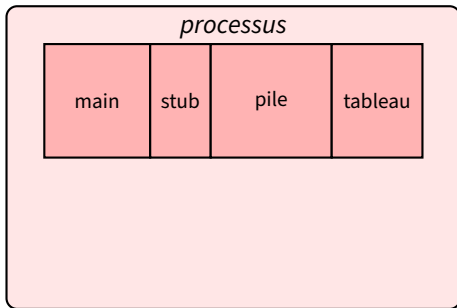
- Pile
- Table des symboles
- Programme principal
- Fonction
- Bibliothèque
- ...

STRUCTURE EN SEGMENTS



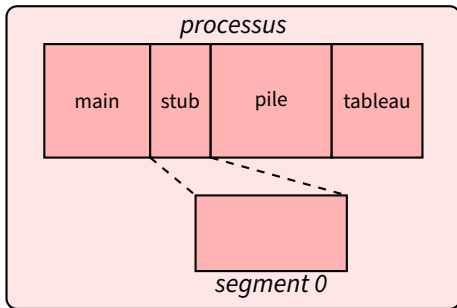
Espace d'adressage logique

STRUCTURE EN SEGMENTS



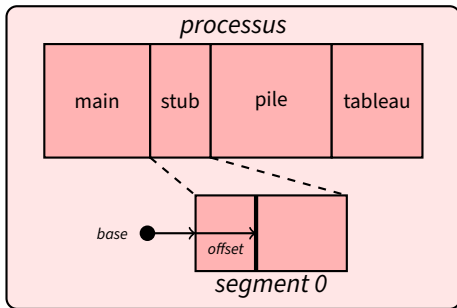
Espace d'adressage logique

STRUCTURE EN SEGMENTS



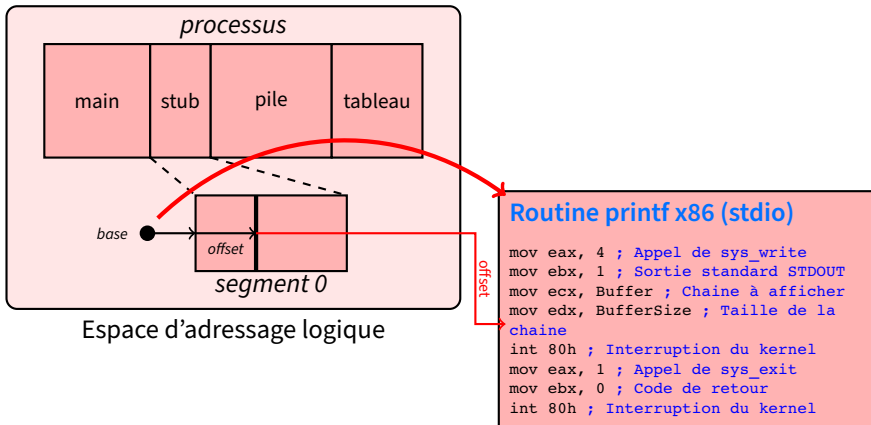
Espace d'adressage logique

STRUCTURE EN SEGMENTS



Espace d'adressage logique

STRUCTURE EN SEGMENTS



CONSTRUCTION DES SEGMENTS

CONSTRUCTION DES SEGMENTS

- À la compilation

- 👉 dépendant du langage
- 👉 chaque segment est référencé par un numéro
 - [table des segments](#) (1 par processus)

CONSTRUCTION DES SEGMENTS

- À la compilation

- 👉 dépendant du langage
- 👉 chaque segment est référencé par un numéro
 - [table des segments](#) (1 par processus)

- Exemple : Java

- Méthodes
- Tas (fonctions, attributs *static*)
- Pile (1 pour chaque thread)
- Class loader

CONSTRUCTION DES SEGMENTS

- À la compilation

- 👉 dépendant du langage
- 👉 chaque segment est référencé par un numéro

→ [table des segments](#) (1 par processus)

- Exemple : Java

- Méthodes
- Tas (fonctions, attributs *static*)
- Pile (1 pour chaque thread)
- Class loader

- Exemple : C

- Variables globales (tas)
- Fonctions de bibliothèques (1 par bibliothèque)
- Programme principal et pile

ADRESSAGE

ADRESSAGE

- Adresse logique

👉 Numéro de segment (*selecteur*) + décalage

ADRESSAGE

- Adresse logique

- 👉 Numéro de segment (*selecteur*) + décalage

- Résolution → Memory Management Unit

- Transformer segment + décalage en adresse physique

ADRESSAGE

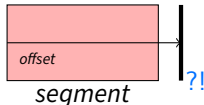
- Adresse logique

👉 Numéro de segment (*selecteur*) + décalage

- Résolution → Memory Management Unit

- Transformer segment + décalage en adresse physique
- Vérifier que le décalage ne sort pas du segment

→ *erreur de segmentation*



RÉSOLUTION D'ADRESSE

RÉSOLUTION D'ADRESSE

- Principe
 - Géré au niveau de la MMU
 - Numéro de segment (*selecteur*) → adresse de base
 - Base + décalage → adresse physique
 - Erreurs de segmentations

RÉSOLUTION D'ADRESSE

- Principe
 - Géré au niveau de la MMU
 - Numéro de segment (*selecteur*) → adresse de base
 - Base + décalage → adresse physique
 - Erreurs de segmentations
- **Table des segments** : pour chaque segment :
 - Base = **adresse physique** de départ
 - Limite = **taille** du segment

...aussi appelée : table des **descripteurs**

RÉSOLUTION D'ADRESSE

RAM

UC

Memory Management Unit

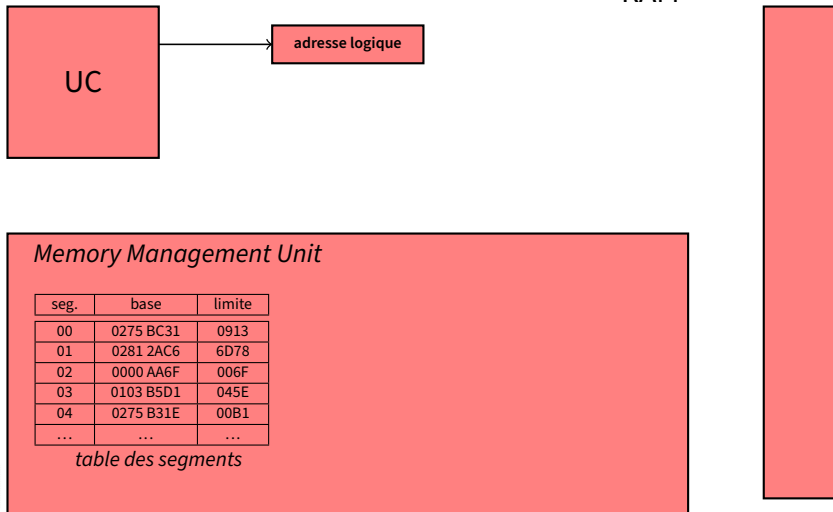
seg.	base	limite
00	0275 BC31	0913
01	0281 2AC6	6D78
02	0000 AA6F	006F
03	0103 B5D1	045E
04	0275 B31E	00B1
...

table des segments



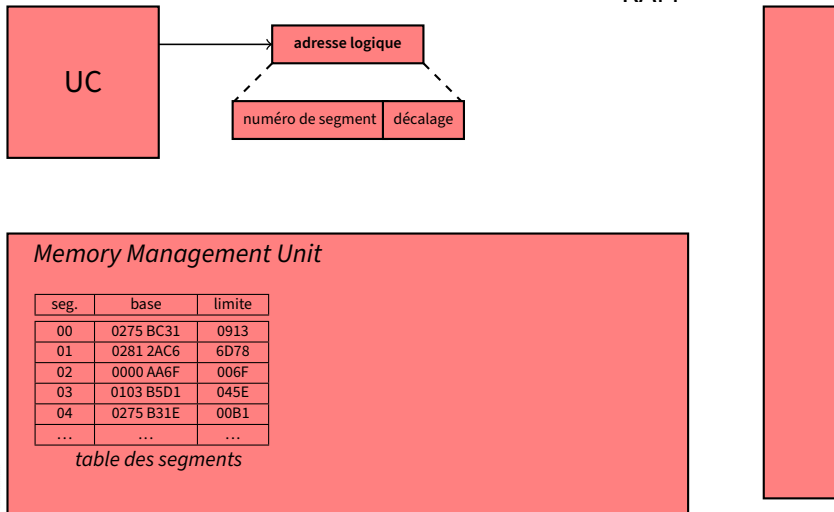
RÉSOLUTION D'ADRESSE

RAM



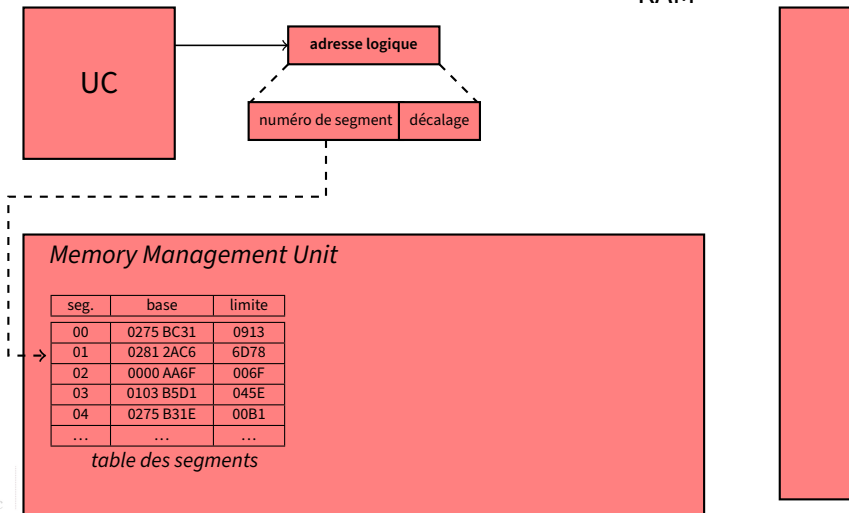
RÉSOLUTION D'ADRESSE

RAM



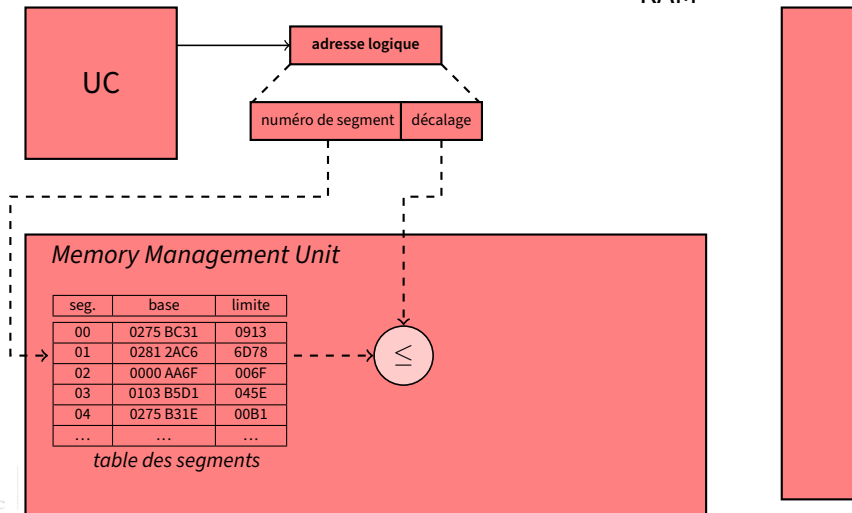
RÉSOLUTION D'ADRESSE

RAM



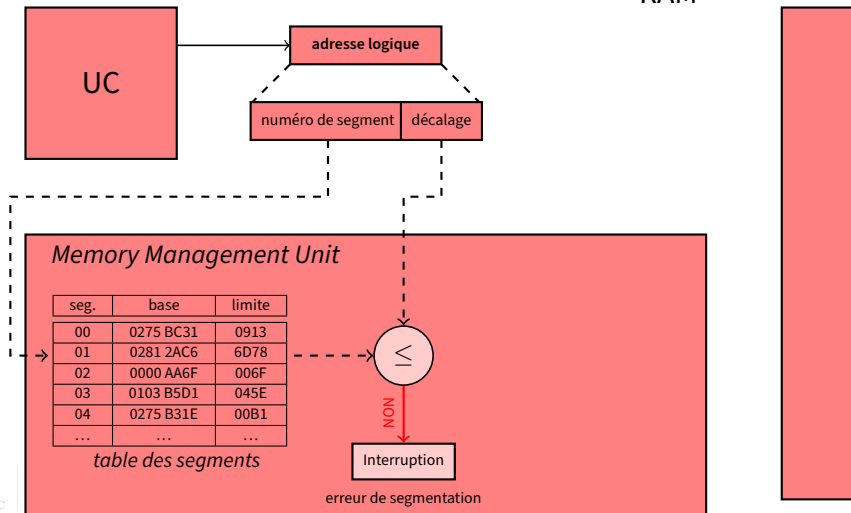
RÉSOLUTION D'ADRESSE

RAM



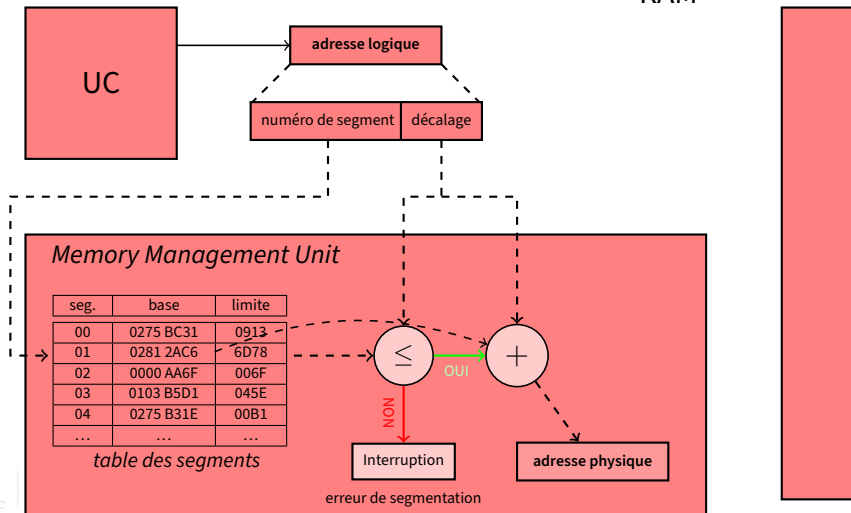
RÉSOLUTION D'ADRESSE

RAM

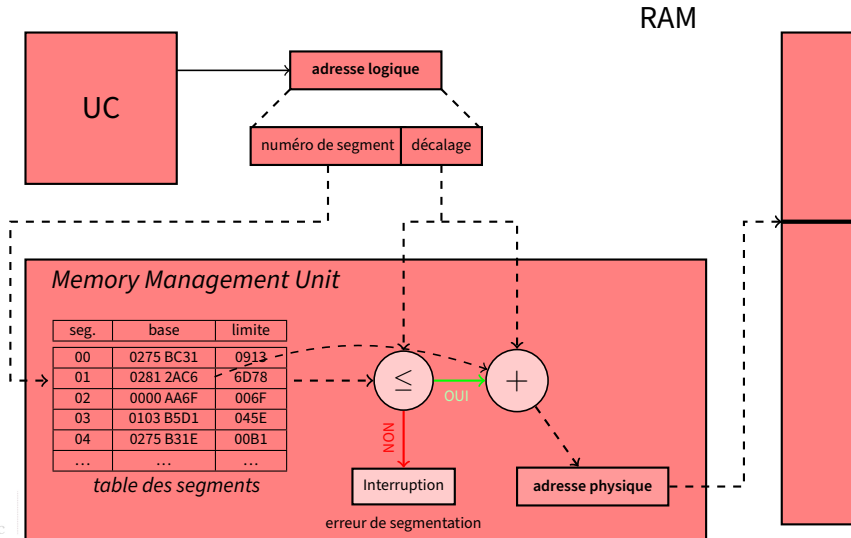


RÉSOLUTION D'ADRESSE

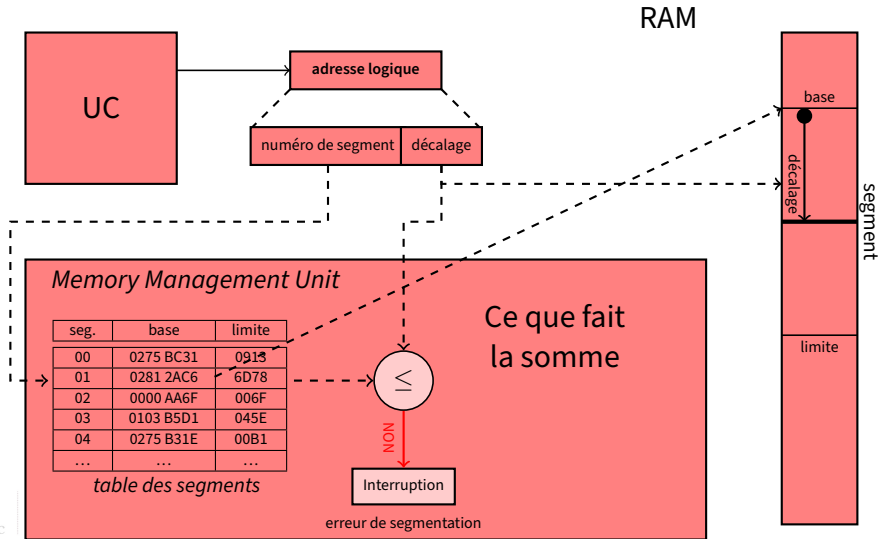
RAM



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



PARTAGE DE SEGMENTS

PARTAGE DE SEGMENTS

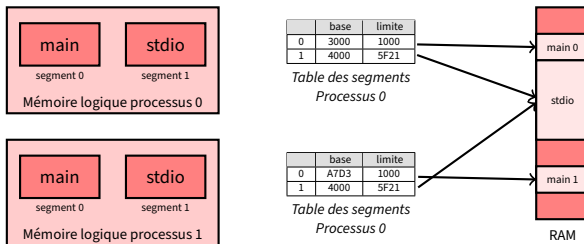
- Un même segment (de code) utilisé par plusieurs processus

PARTAGE DE SEGMENTS

- Un même segment (de code) utilisé par plusieurs processus
- **Exemple** : bibliothèques
 - 1 segment pour la bibliothèque
 - N processus utilisent le même segment

PARTAGE DE SEGMENTS

- Un même segment (de code) utilisé par plusieurs processus
- **Exemple** : bibliothèques
 - 1 segment pour la bibliothèque
 - N processus utilisent le **même segment**



PARTAGE DE SEGMENTS

PARTAGE DE SEGMENTS

- **Résolution d'adresse** des segments partagés
 - Appel de routine = saut d'adresse
 - Utilisation de *stub* : code remplaçable
- Adresse définie lors du **premier** chargement

PARTAGE DE SEGMENTS

- **Résolution d'adresse** des segments partagés

- Appel de routine = saut d'adresse
- Utilisation de *stub* : code remplaçable

→ Adresse définie lors du **premier** chargement

- ✓ **Protection**

👉 Par segment **et** par processus!

- Les segments peuvent être marqué **Read**, **Write** ou **ReadWrite**.

→ bit de protection **dans la table des segments**

PARTAGE DE SEGMENTS

- **Résolution d'adresse** des segments partagés

- Appel de routine = saut d'adresse
- Utilisation de *stub* : code remplaçable

→ Adresse définie lors du **premier** chargement

- ✓ **Protection**



Par segment **et** par processus!

- Les segments peuvent être marqué **Read**, **Write** ou **ReadWrite**.

→ bit de protection **dans la table des segments**

- ✓ **Partage de code**

- ✓ Bibliothèques, processus multi-utilisateurs



- Plus difficile qu'avec la pagination (stub)



- ...mais résolu à la compilation + chargement par l'OS

QUELQUES LIMITES

QUELQUES LIMITES

✗ Taille des segments variables



On retombe sur le problème de l'allocation contigüe :

- Fragmentation → perte de mémoire
- Défragmentation → perte de temps à l'exécution

QUELQUES LIMITES

✗ Taille des segments variables



On retombe sur le problème de l'allocation contigüe :

- Fragmentation → perte de mémoire
- Défragmentation → perte de temps à l'exécution

✓ Solution → **Segmentation avec pagination**

SEGMENTATION AVEC PAGINATION

SEGMENTATION AVEC PAGINATION

- Paginer les segments
 - 👉 Réduit la fragmentation et les problème d'allocation
 - 👉 Permet le partage et l'adressage des segments

SEGMENTATION AVEC PAGINATION

- Paginer les segments
 - 👉 Réduit la fragmentation et les problème d'allocation
 - 👉 Permet le partage et l'adressage des segments
- Adresse logique = sélecteur + décalage₁
résolu par segmentation donne :

SEGMENTATION AVEC PAGINATION

- Paginer les segments
 - 👉 Réduit la fragmentation et les problème d'allocation
 - 👉 Permet le partage et l'adressage des segments
- Adresse logique = sélecteur + décalage₁
résolu par segmentation donne :
- Adresse **linéaire** = (répertoire +)¹ page + décalage₂

SEGMENTATION AVEC PAGINATION

- Paginer les segments
 - 👉 Réduit la fragmentation et les problème d'allocation
 - 👉 Permet le partage et l'adressage des segments
- Adresse logique = sélecteur + décalage₁
résolu par segmentation donne :
- Adresse **linéaire** = (répertoire +)¹ page + décalage₂
- Adresse physique = cadre de page + décalage₂

SEGMENTATION AVEC PAGINATION

- Paginer les segments
 - 👉 Réduit la fragmentation et les problème d'allocation
 - 👉 Permet le partage et l'adressage des segments
- Adresse logique = sélecteur + décalage₁
résolu par segmentation donne :
- Adresse **linéaire** = (répertoire +)¹ page + décalage₂
- Adresse physique = cadre de page + décalage₂
- **Mémory Management Unit**
 - Table des **descripteurs** : adresse logique → adresse linéaire
 - Répertoire¹ + table des pages : adresse linéaire → physique

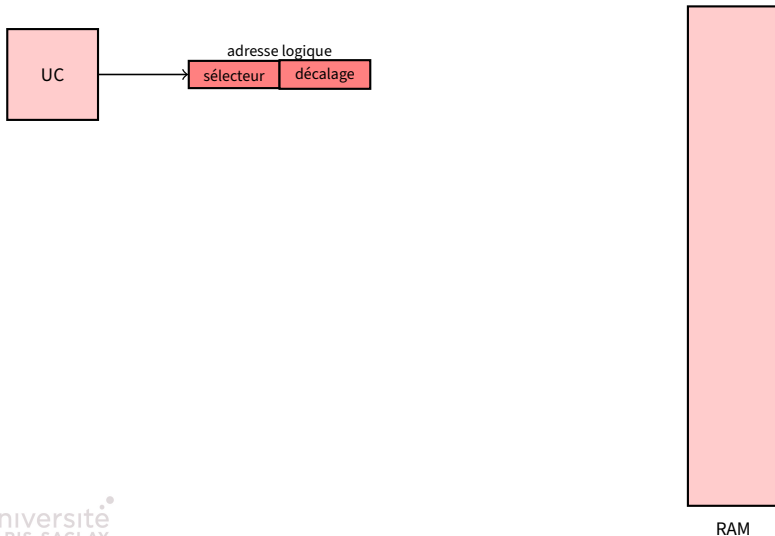
¹ si pagination à 2 niveaux

IMPLÉMENTATION

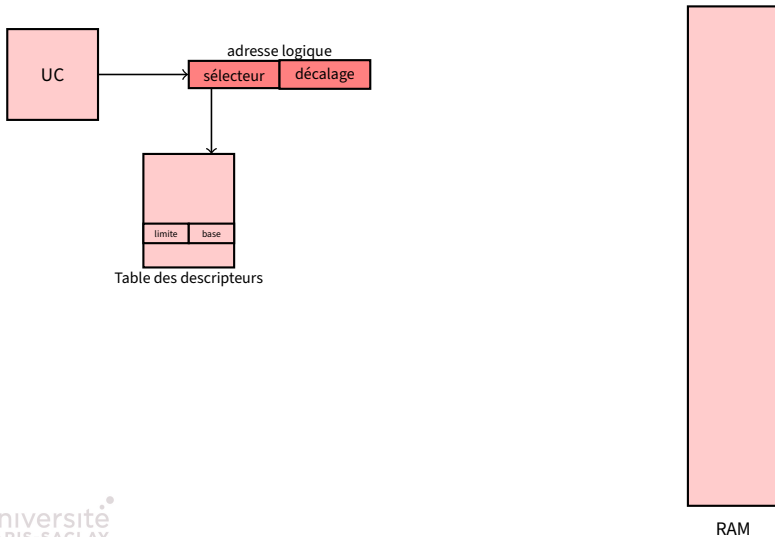


RAM

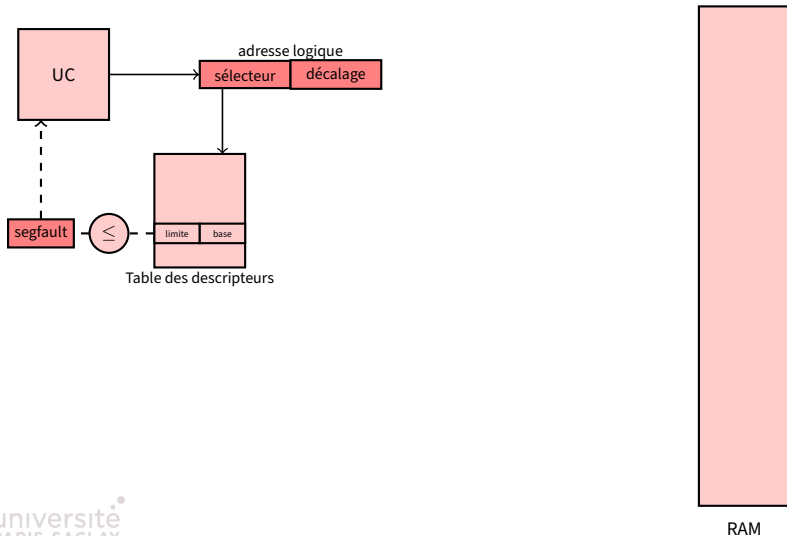
IMPLÉMENTATION



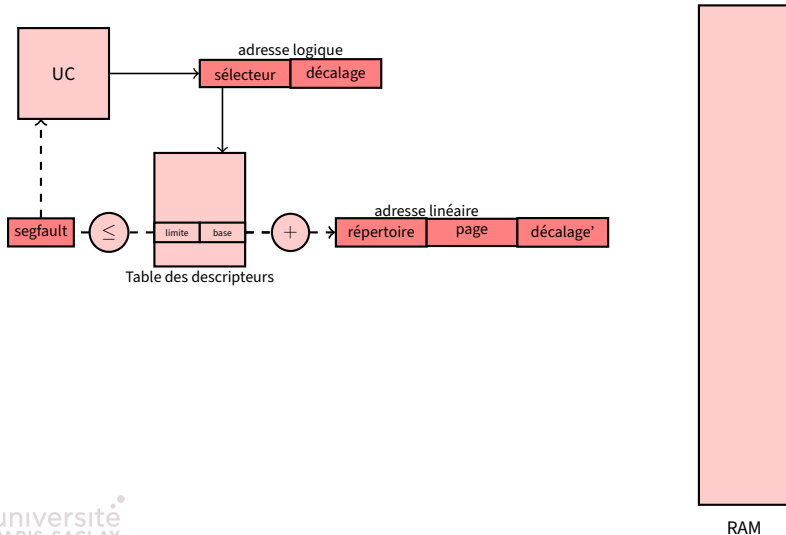
IMPLÉMENTATION



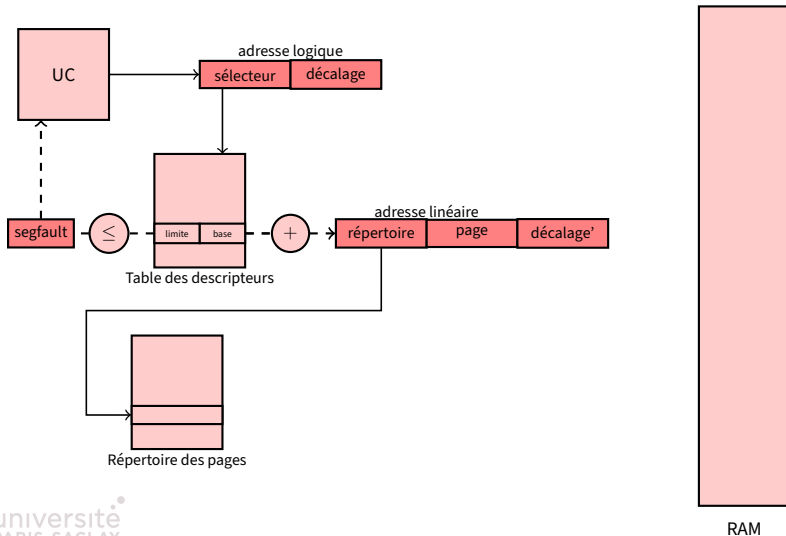
IMPLÉMENTATION



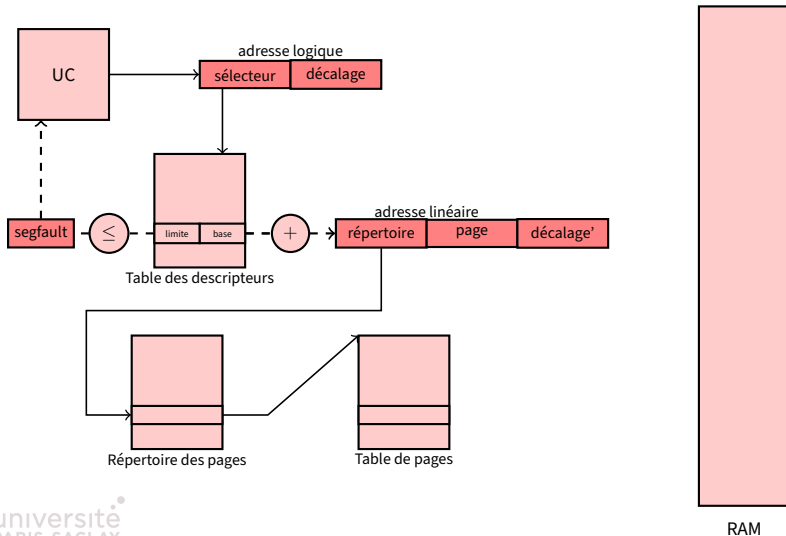
IMPLÉMENTATION



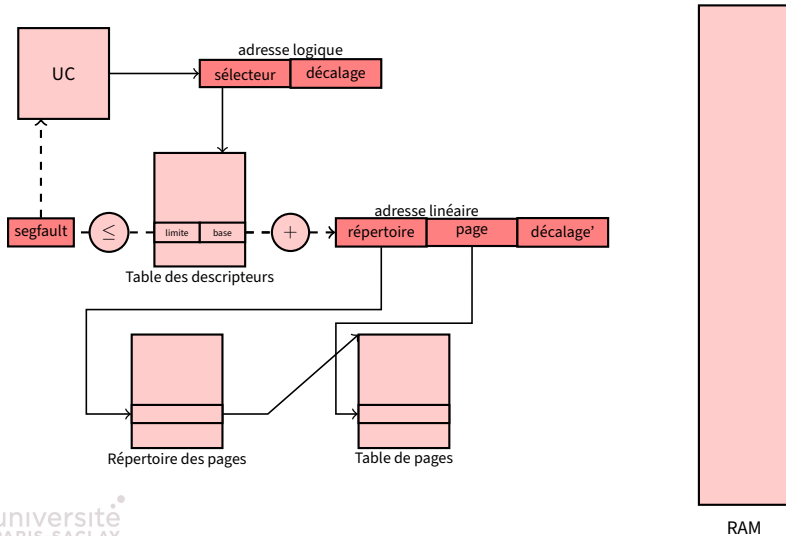
IMPLÉMENTATION



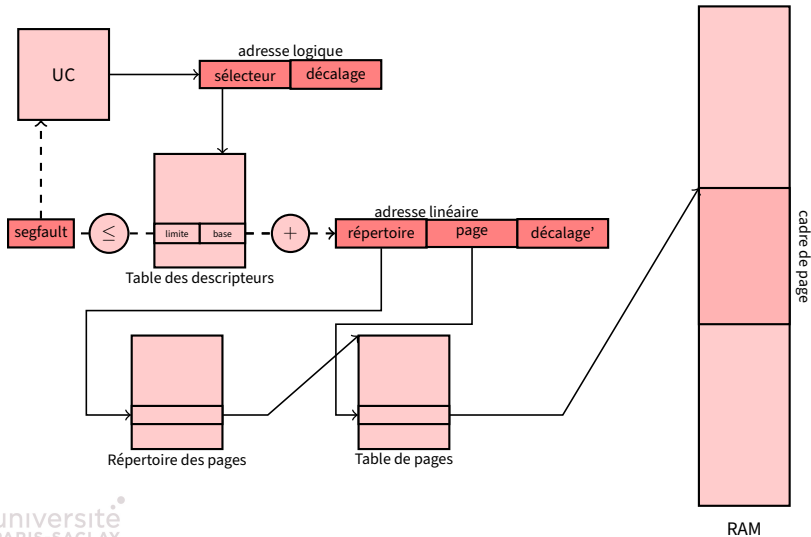
IMPLÉMENTATION



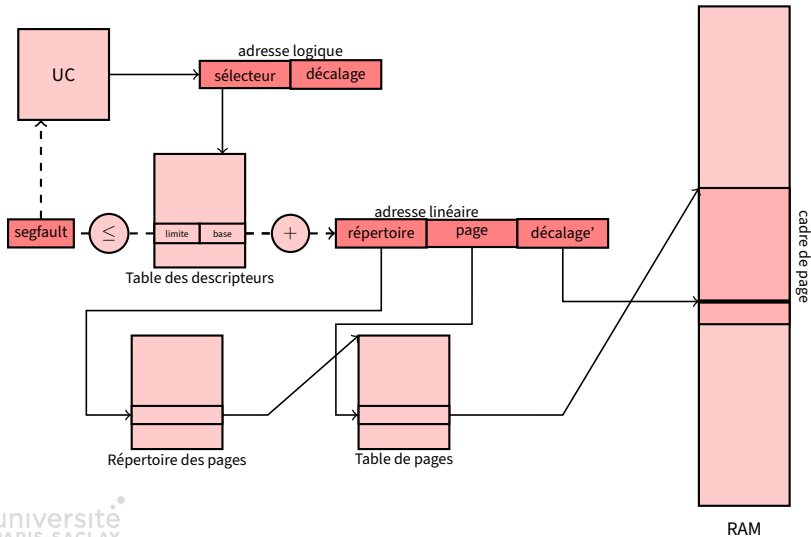
IMPLÉMENTATION



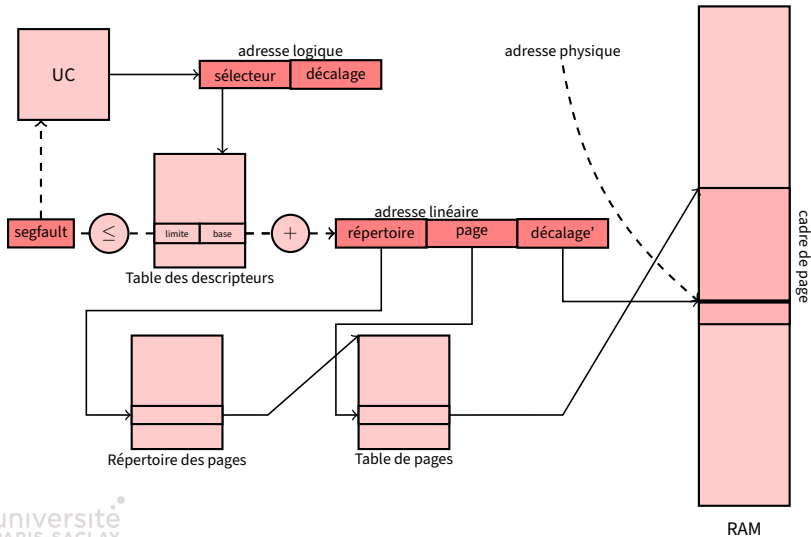
IMPLÉMENTATION



IMPLÉMENTATION



IMPLÉMENTATION



PLAN

- La mémoire
- La mémoire virtuelle
- La mémoire segmentée
- Synthèse

[Retour à l'accueil](#) - [Retour au plan](#)

SYNTHÈSE

SYNTHÈSE

- Gestion de la mémoire

SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions

SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions
- La mémoire paginée

SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions
- La mémoire paginée
- La mémoire virtuelle

SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions
- La mémoire paginée
- La mémoire virtuelle
- La mémoire segmentée

SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions
- La mémoire paginée
- La mémoire virtuelle
- La mémoire segmentée
- La Segmentation avec pagination

THE END

[Retour à l'accueil](#) - [Retour au plan](#)