Laboratoire Méthodes Formelles

CentraleSupélec | université PARIS-SACLAY

LES SYSTÈMES D'EXPLOITATION

ORGANISATION ET GESTION DE LA MÉMOIRE

3A - Cursus Ingénieurs - Dominante Informatique et Numérique
CentraleSupelec - Université Paris-Saclay - 2025/2026

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

OUTLINE

Back to the begin - Back to the outline

OUTLINE

Back to the begin - Back to the outline

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)
  - ...

CentraleSupélec
université PARIS-SACLAY

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 (**un espace d'adresses**).

LA MÉMOIRE POUR QUI ET POURQUOI ?

Pour les processus

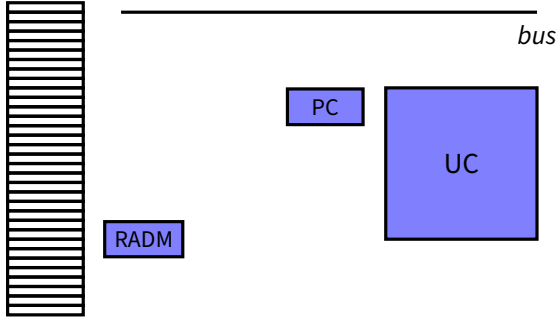
- A la création d'un processus, l'**OS crée un PCB** et alloue de la mémoire pour le processus.
- Pour des raisons de **sécurité**, chaque **processus** doit utiliser une zone mémoire distincte (**un espace d'adresses**).
  - quel mécanisme d'**allocation** de cet espace ?
  - comment assurer la **protection**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

CentraleSupélec
université PARIS-SACLAY

ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

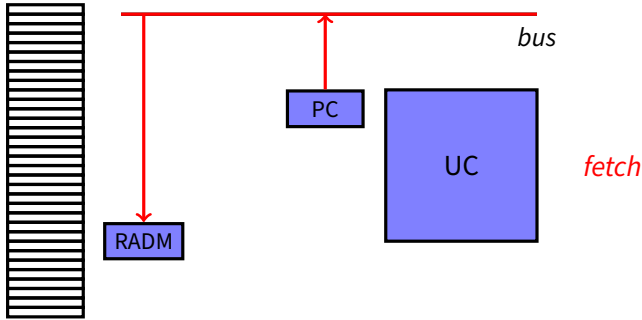
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des données $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

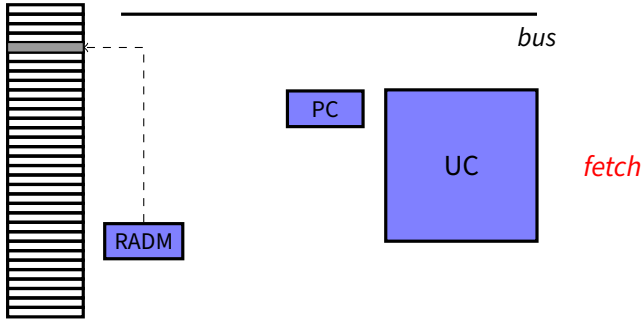
- Des **instructions** $\longrightarrow$ registre **PC** sur le processeur
- Des données $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

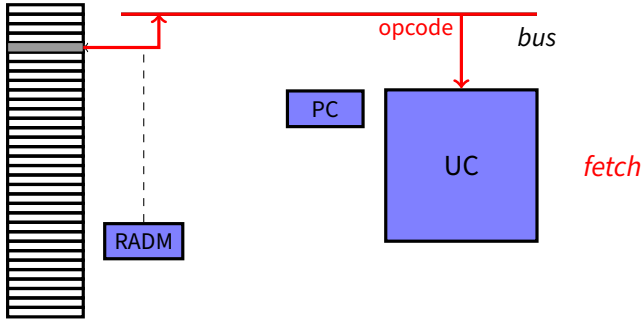
- Des **instructions** $\longrightarrow$ registre **PC** sur le processeur
- Des données $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

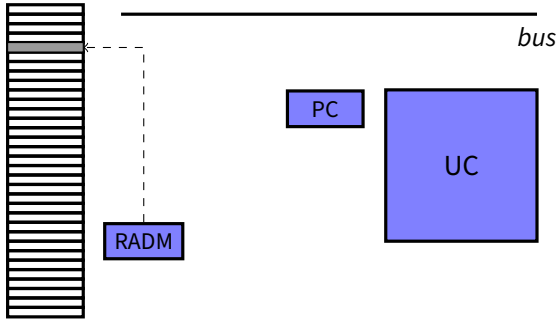
- Des **instructions** → registre **PC** sur le processeur
- Des données → registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

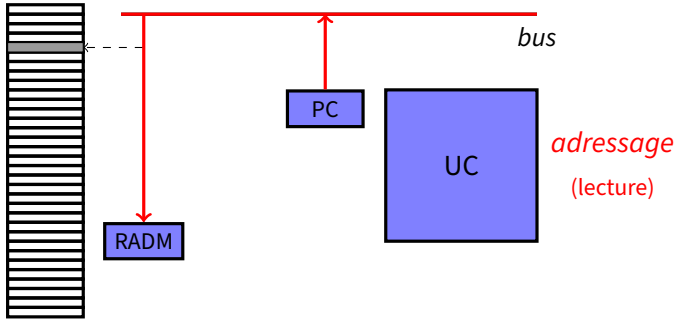
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des données $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

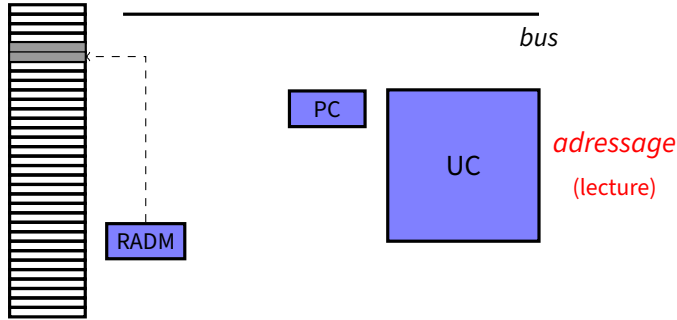
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

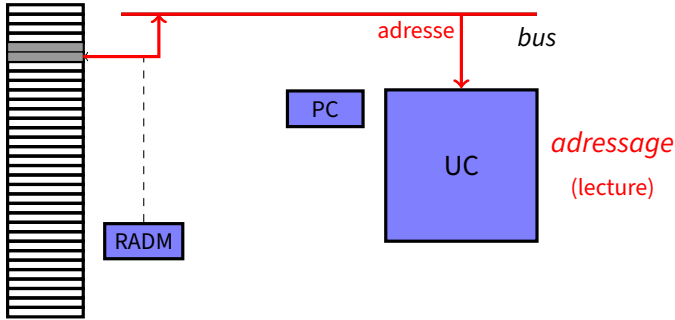
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

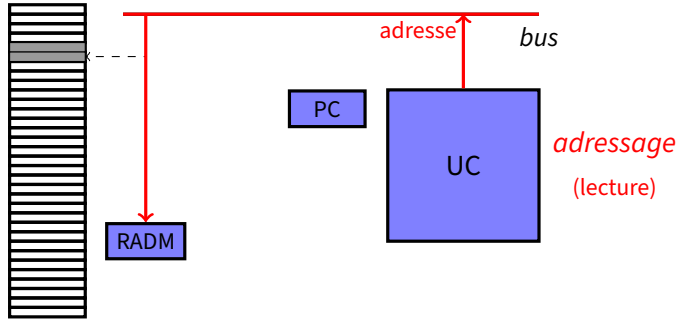
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

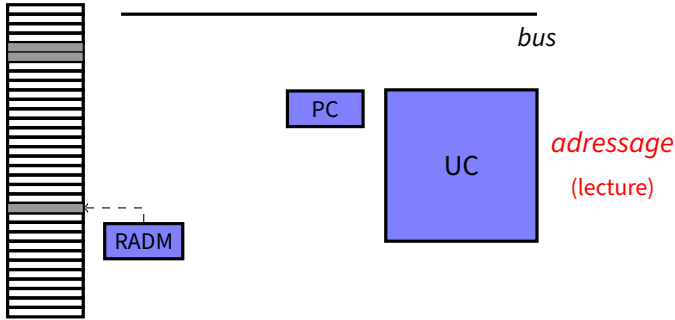
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

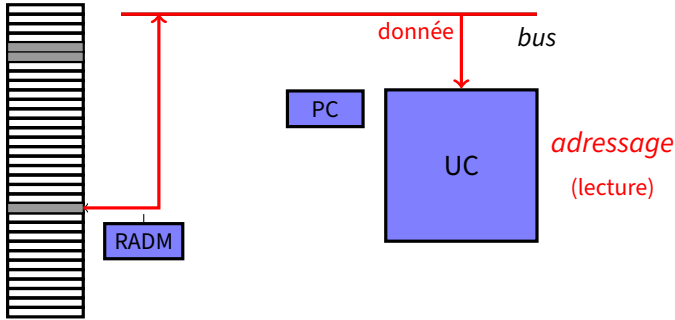
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

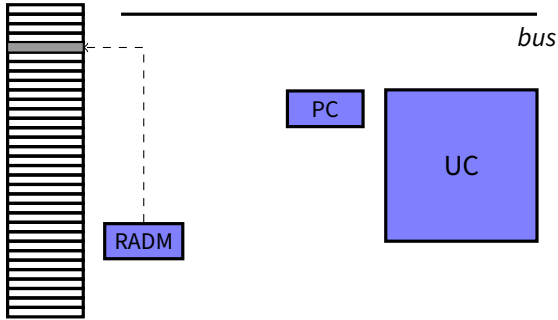
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

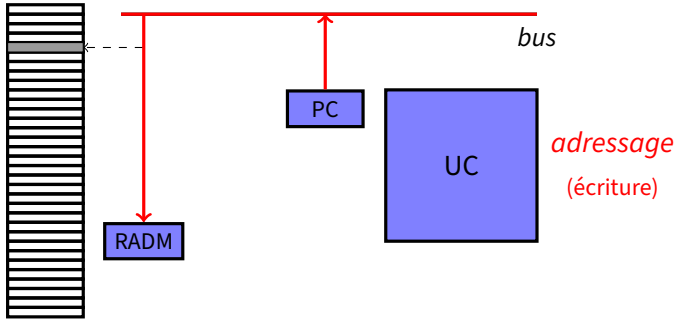
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

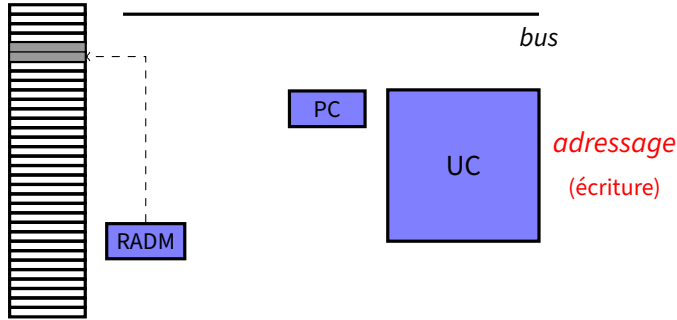
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

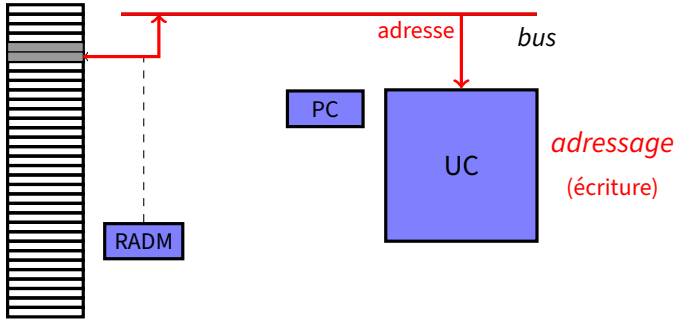
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

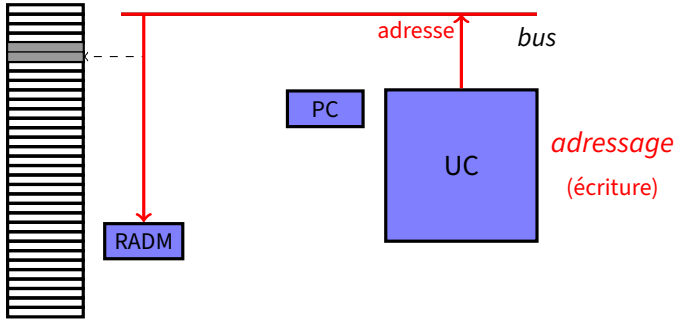
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

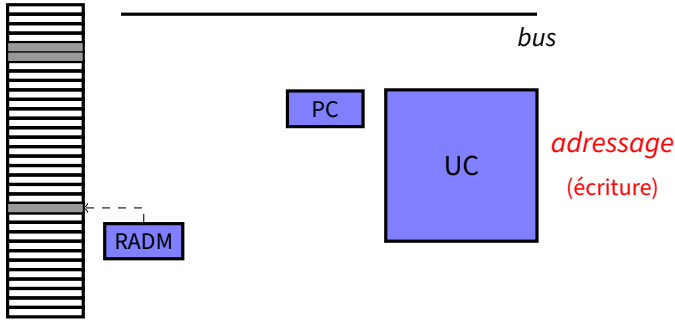
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

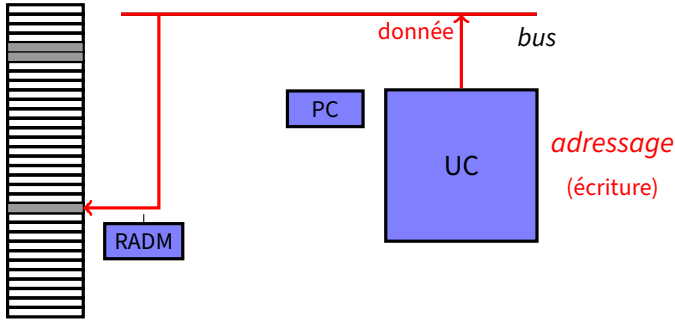
- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ registre **RADM** sur le processeur



ESPACE DE STOCKAGE

Ensemble ordonné de **cases** indexée par leur **adresse** (numéro de la case) et contenant :

- Des instructions $\longrightarrow$ registre **PC** sur le processeur
- Des **données** $\longrightarrow$ 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

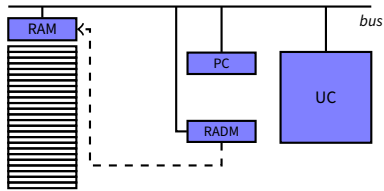
STRUCTURE DE LA MÉMOIRE

- 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 $00 \ldots 0$ à $11 \ldots 1$

STRUCTURE DE LA MÉMOIRE

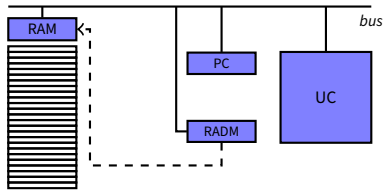
- Chaque **case mémoire** est associée à une **adresse**
  - le numéro de la case
- Cette **adresse** est obtenue depuis une instruction
  - l'adresse est en **binaire** sur n bits
- Il y a donc 2^n différentes adresses possibles
  - 2^n cases de $00 \ldots 0$ à $11 \ldots 1$
- **Exemple** : 32 bits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

FONCTIONNEMENT DE LA MÉMOIRE



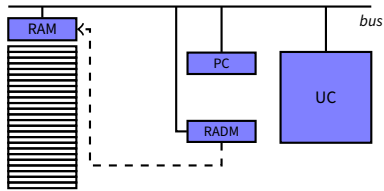
FONCTIONNEMENT DE LA MÉMOIRE

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



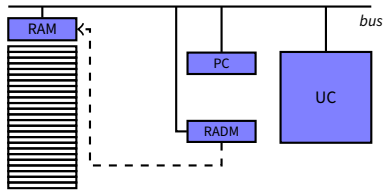
FONCTIONNEMENT DE LA MÉMOIRE

- 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**.

CentraleSupélec
université PARIS-SACLAY

MÉTHODES DE LIAISON D'ADRESSES

MÉTHODES DE LIAISON D'ADRESSES

- **À la compilation** $\longrightarrow$ 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** $\longrightarrow$ 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** $\longrightarrow$ la taille du processus est fixée
  - **adresses virtuelles** (**logiques**) : adresses utilisées dans le programme
  - **adresses physiques** : adresses utilisées dans la **RAM**

MÉTHODES DE LIAISON D'ADRESSES

- **À la compilation** $\longrightarrow$ 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

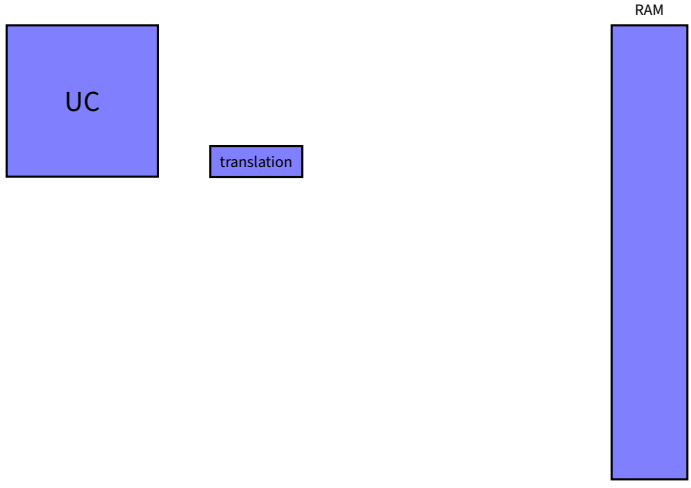
- **Au chargement** $\longrightarrow$ la taille du processus est fixée
  - **adresses virtuelles** (**logiques**) : adresses utilisées dans le programme
  - **adresses physiques** : adresses utilisées dans la **RAM**
  - nécessite un *composant de translation* dans l'**UC** (**Memory Management Unit - MMU**)

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** (**logiques**) : adresses utilisées dans le programme
  - **adresses physiques** : adresses utilisées dans la **RAM**
  - nécessite un *composant de translation* dans l'**UC** (**Memory Management Unit - MMU**)
- **À l'exécution** → le processus a besoin de plus de place
  - déplacer le processus + réédition de lien

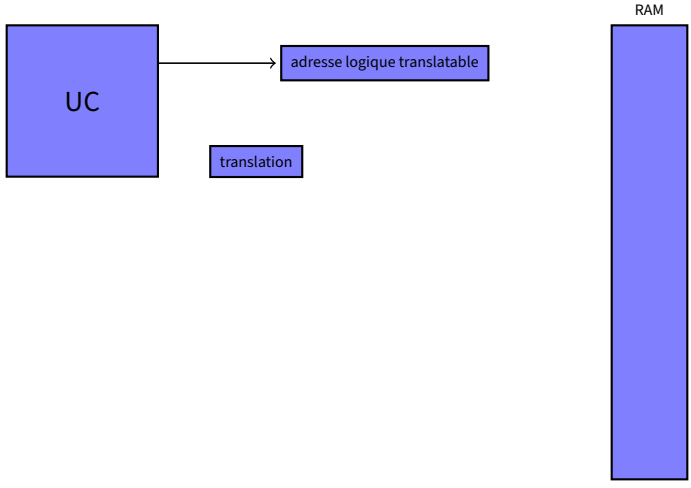
RÉSOLUTION D'ADRESSE

MEMORY MANAGEMENT UNIT - MMU



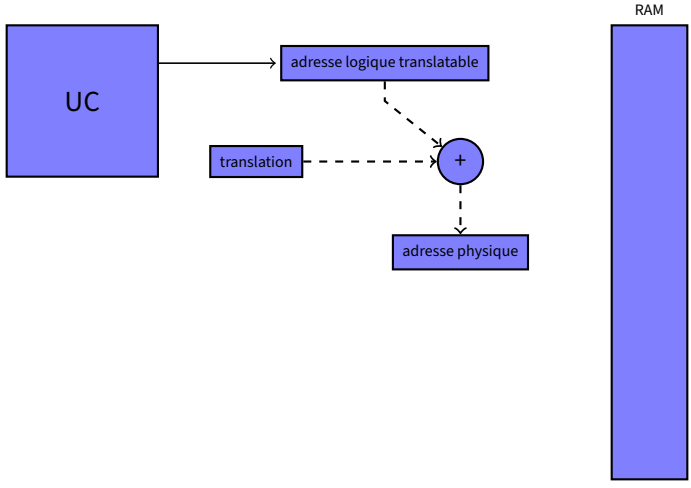
RÉSOLUTION D'ADRESSE

MEMORY MANAGEMENT UNIT - MMU



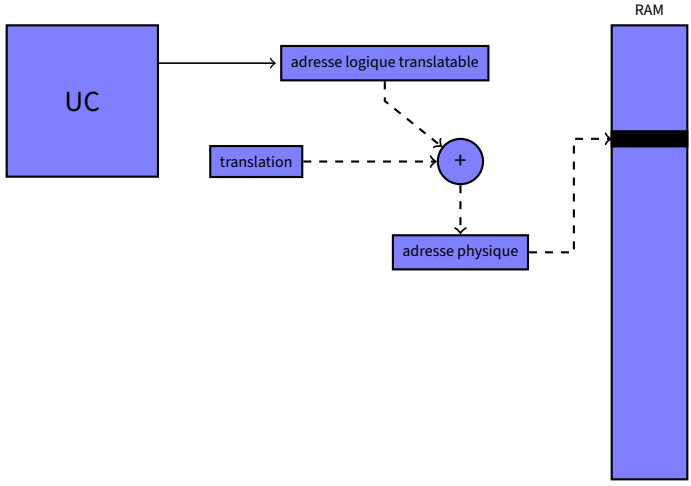
RÉSOLUTION D'ADRESSE

MEMORY MANAGEMENT UNIT - MMU



RÉSOLUTION D'ADRESSE

MEMORY MANAGEMENT UNIT - MMU



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**)

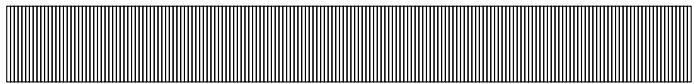
STRATÉGIES D'ALLOCATION DE MÉMOIRE

- 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

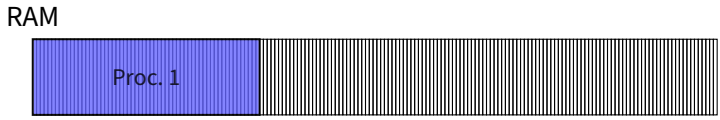
Les processus constituent **un seul bloc** non décomposable.

RAM



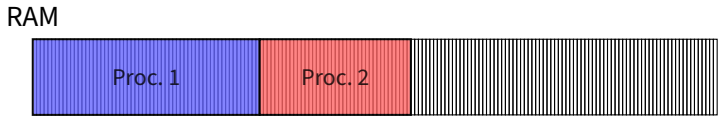
ALLOCATION PAR PARTITION

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



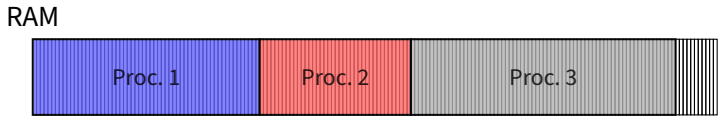
ALLOCATION PAR PARTITION

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



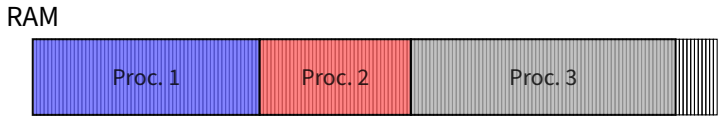
ALLOCATION PAR PARTITION

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



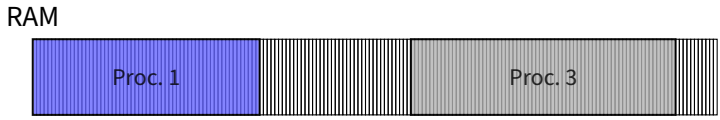
ALLOCATION PAR PARTITION

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



ALLOCATION PAR PARTITION

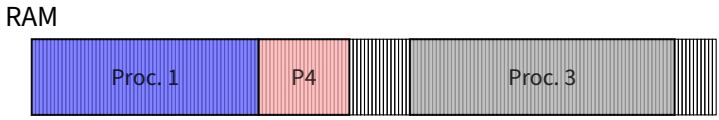
Les processus constituent **un seul bloc** non décomposable.



✘ des trous apparaissent $\longrightarrow$ **fragmentation**

ALLOCATION PAR PARTITION

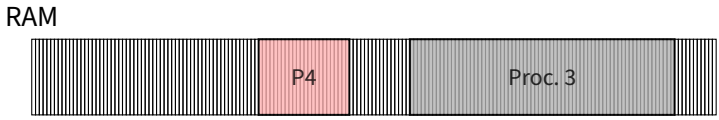
Les processus constituent **un seul bloc** non décomposable.



✘ des trous apparaissent $\longrightarrow$ **fragmentation**

ALLOCATION PAR PARTITION

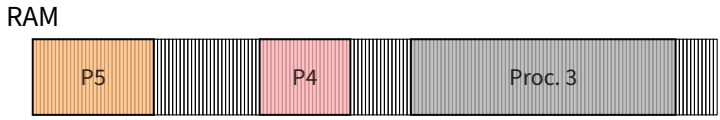
Les processus constituent **un seul bloc** non décomposable.



✘ des trous apparaissent $\longrightarrow$ **fragmentation**

ALLOCATION PAR PARTITION

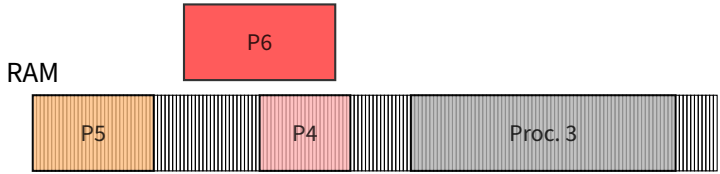
Les processus constituent **un seul bloc** non décomposable.



✘ des trous apparaissent $\longrightarrow$ **fragmentation**

ALLOCATION PAR PARTITION

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

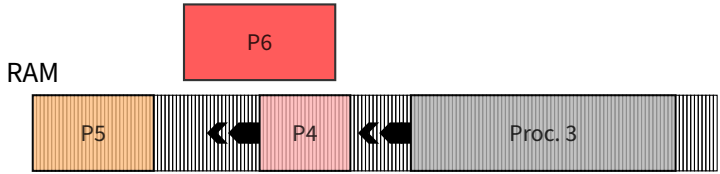


✘ des trous apparaissent → **fragmentation**

✘ les gros processus ne peuvent pas rentrer

ALLOCATION PAR PARTITION

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

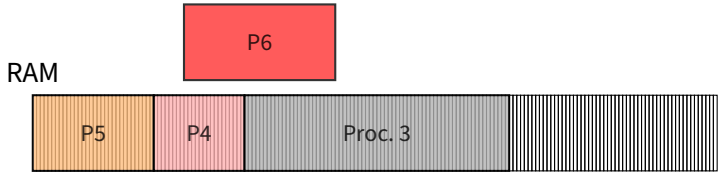


✘ des trous apparaissent → **fragmentation**

✘ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

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

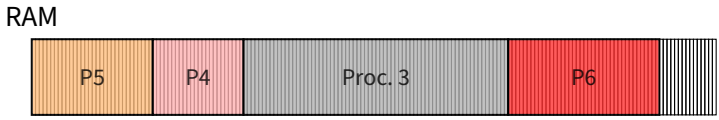


✘ des trous apparaissent → **fragmentation**

✘ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

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

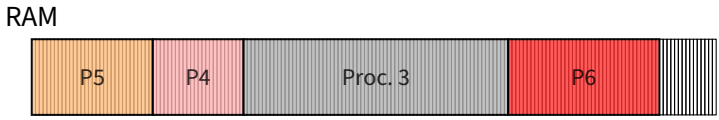


✘ des trous apparaissent → **fragmentation**

✘ les gros processus ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

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



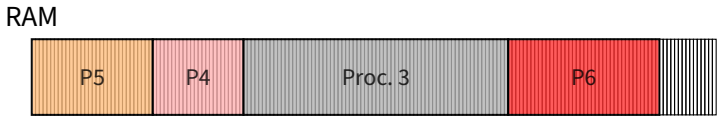
✘ des trous apparaissent ⟶ **fragmentation**

✘ les gros processus ne peuvent pas rentrer ⟶ **défragmenter**

➠ réduire la **fragmentation**

ALLOCATION PAR PARTITION

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



- ✘ 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** $\longrightarrow$ choisir dans quelle *zone libre* placer un processus

ALLOCATION PAR PARTITION

- **Stratégies d'allocation** $\longrightarrow$ choisir dans quelle *zone libre* placer un processus

  ➠ **First Fit** : premier bloc libre

ALLOCATION PAR PARTITION

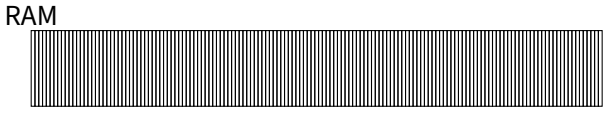
- **Stratégies d'allocation** $\longrightarrow$ choisir dans quelle *zone libre* placer un processus
  - **First Fit** : premier bloc libre
  - **Best Fit** : plus petit bloc libre

ALLOCATION PAR PARTITION

- **Stratégies d'allocation** $\longrightarrow$ choisir dans quelle *zone libre* placer un processus

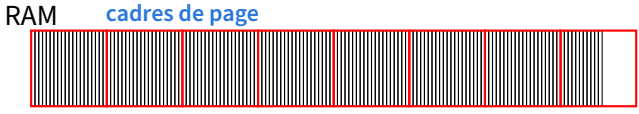
  - **First Fit** : premier bloc libre
  - **Best Fit** : plus petit bloc libre
  - **Worst Fit** : plus grand bloc libre

ALLOCATION PAR PAGINATION



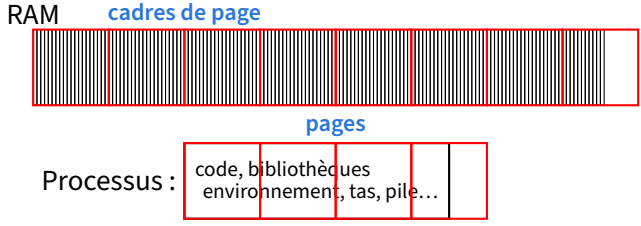
ALLOCATION PAR PAGINATION

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



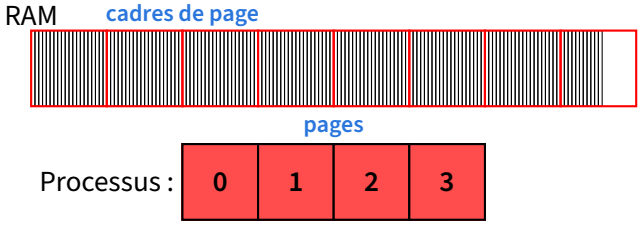
ALLOCATION PAR PAGINATION

- 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



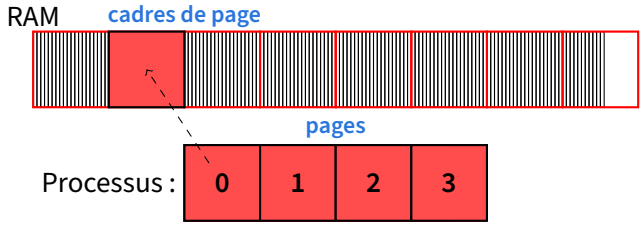
ALLOCATION PAR PAGINATION

- 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**



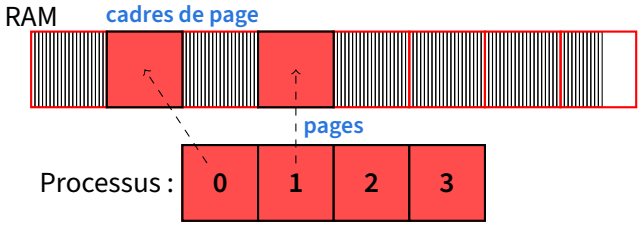
ALLOCATION PAR PAGINATION

- 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



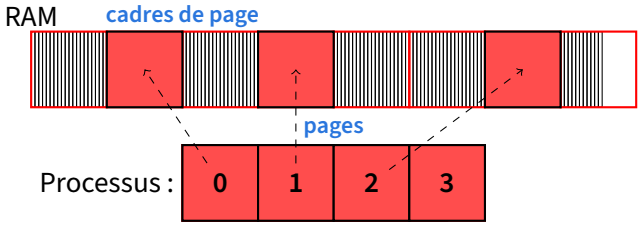
ALLOCATION PAR PAGINATION

- 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



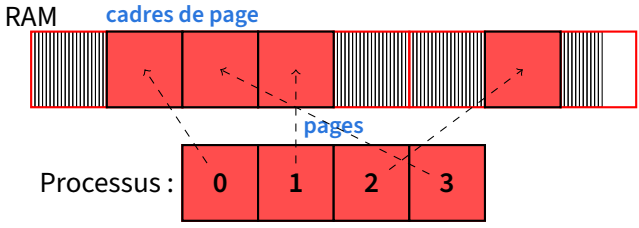
ALLOCATION PAR PAGINATION

- 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 $\longrightarrow$ *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 $\longrightarrow$ *rajouter* des pages
  - pas besoin de le ré-allouer entièrement
- **Mémoire virtuelle** : charger uniquement les pages dont le processus a besoin.

ALLOCATION PAR PAGINATION

ADRESSAGE

ALLOCATION PAR PAGINATION

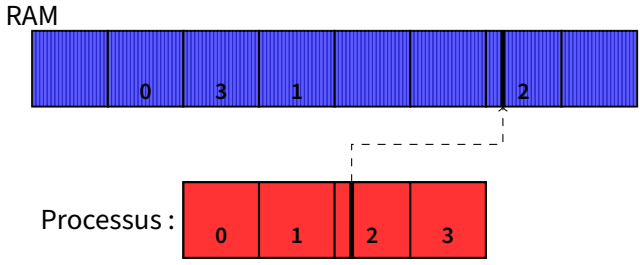
ADRESSAGE

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

ALLOCATION PAR PAGINATION

ADRESSAGE

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



ALLOCATION PAR PAGINATION

ADRESSAGE

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

ALLOCATION PAR PAGINATION

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

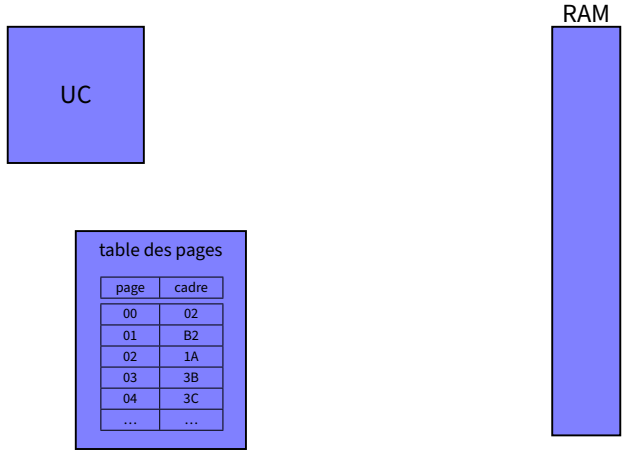
ALLOCATION PAR PAGINATION

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

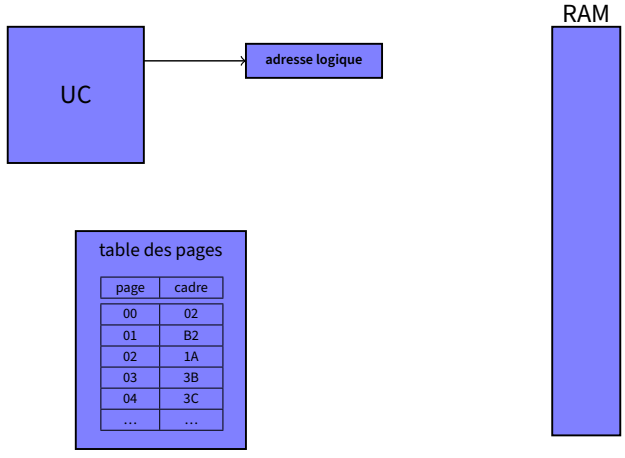
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



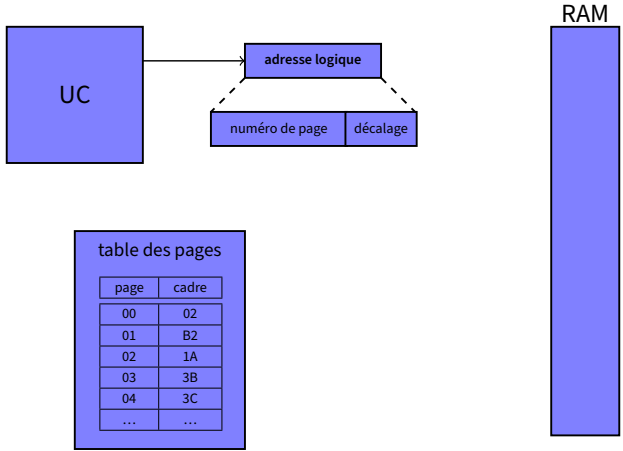
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



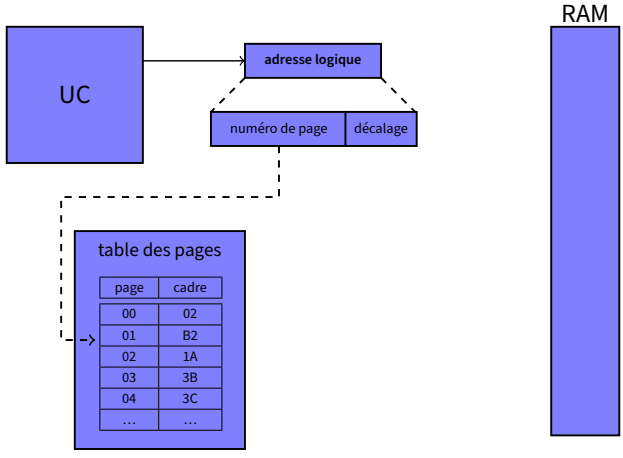
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



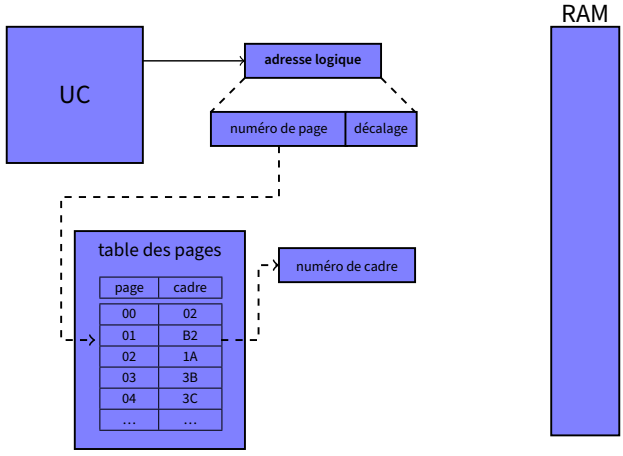
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



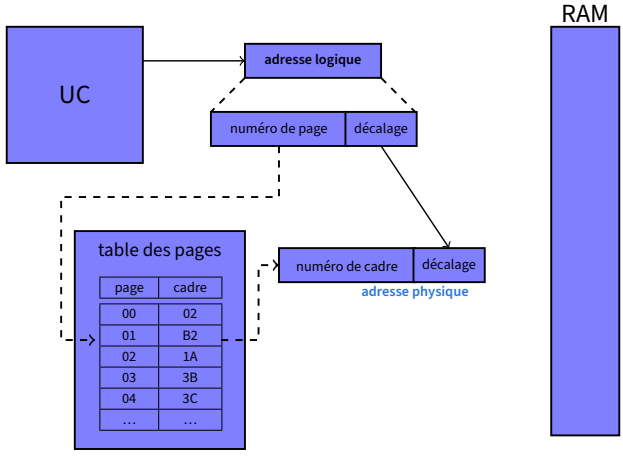
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



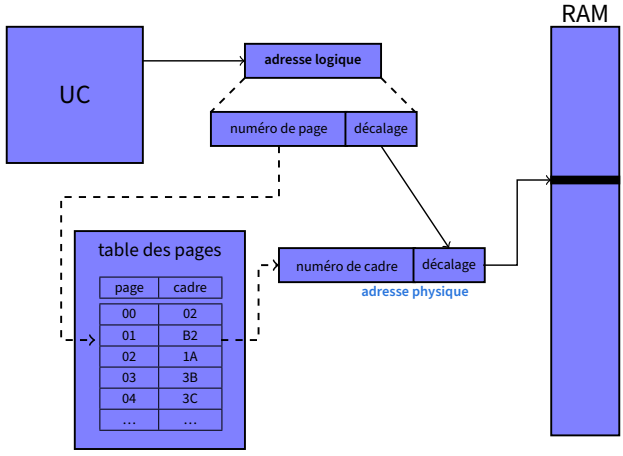
ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE



ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE

ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel $\rightarrow$ la **MMU**

ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE

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

ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel $\rightarrow$ 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 PAR PAGINATION

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel $\rightarrow$ 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

ALLOCATION PAR PAGINATION

RÉSOLUTION D'ADRESSE

- En pratique, géré au niveau matériel $\rightarrow$ 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

ALLOCATION PAR PAGINATION

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 $\longrightarrow$ Grand espace d'adressage (ex : 32 bits)

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

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

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

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

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

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

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
  - Grande table des pages $\rightarrow$ 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 $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
  - Grande table des pages $\rightarrow$ 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 $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

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

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
  - Grande table des pages $\rightarrow$ 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} $\rightarrow$ place mémoire perdue
  - 2^{21} octets $\approx$ 2 Mo par processus

GESTION DE L'ESPACE D'ADRESSAGE

Problème $\rightarrow$ Grand espace d'adressage (ex : 32 bits)

- *Trop de pages* (ex : $n = 20, m = 12$)
  - Grande table des pages $\rightarrow$ 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} $\rightarrow$ place mémoire perdue
  - 2^{21} octets $\approx$ 2 Mo par processus
  - Pas gérable au niveau du MMU

CentraleSupélec
université PARIS-SACLAY

PAGINATION HIÉRARCHIQUE
2 NIVEAUX (OU PLUS)

PAGINATION HIÉRARCHIQUE
2 NIVEAUX (OU PLUS)

- **Solution** $\longrightarrow$ paginer la table des pages

PAGINATION HIÉRARCHIQUE 2 NIVEAUX (OU PLUS)

- **Solution** $\longrightarrow$ paginer la table des pages
  - ne charger que les tables utiles

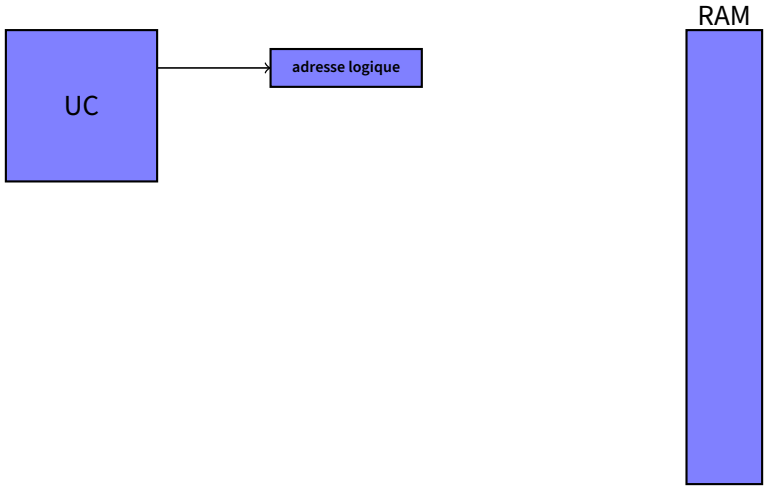
PAGINATION HIÉRARCHIQUE 2 NIVEAUX (OU PLUS)

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

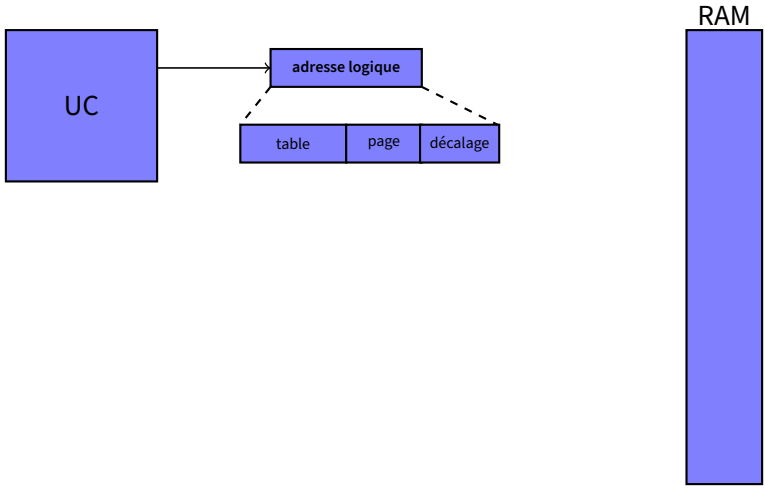
PAGINATION HIÉRARCHIQUE 2 NIVEAUX (OU PLUS)

- **Solution** $\longrightarrow$ 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

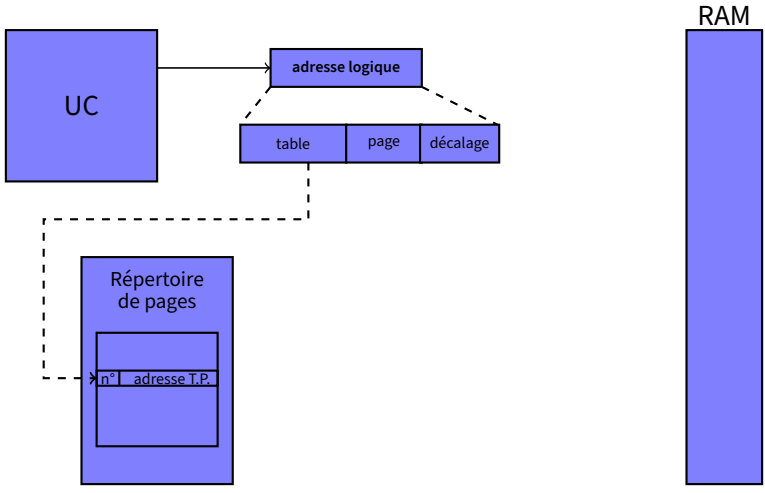
PAGINATION À DEUX NIVEAUX



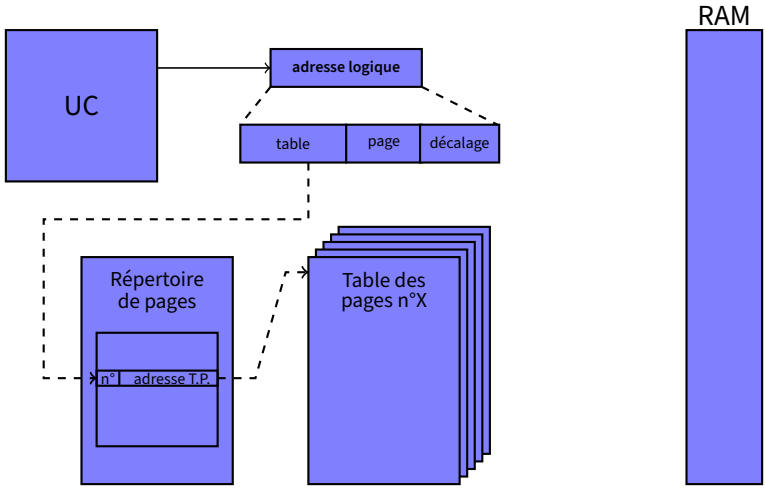
PAGINATION À DEUX NIVEAUX



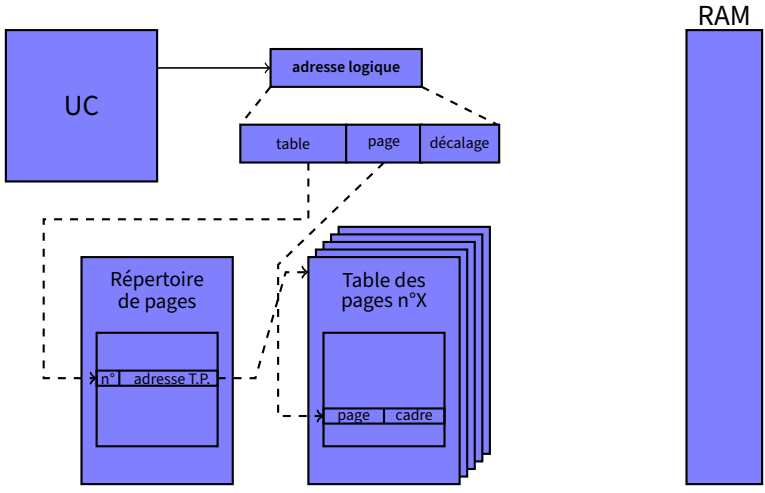
PAGINATION À DEUX NIVEAUX



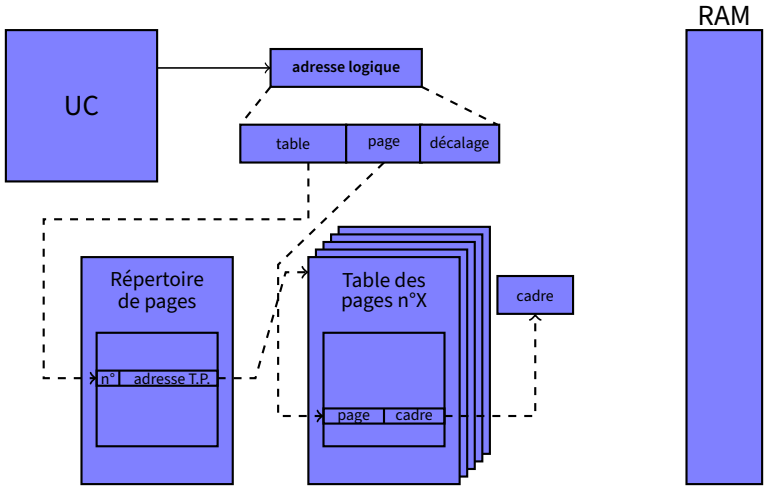
PAGINATION À DEUX NIVEAUX



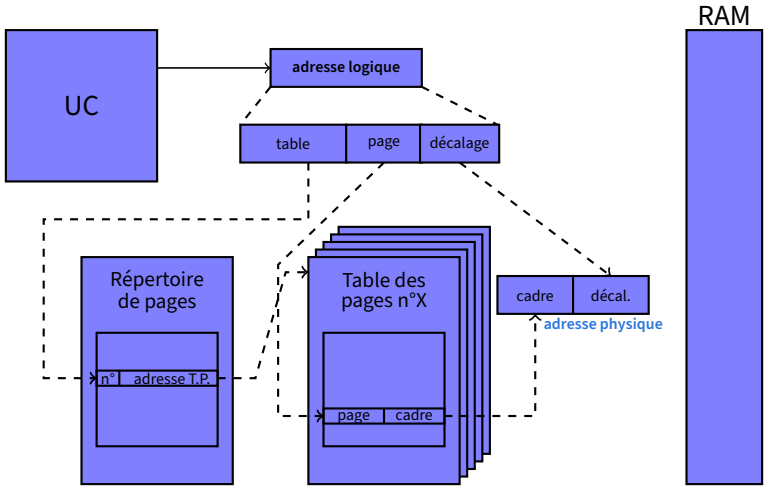
PAGINATION À DEUX NIVEAUX



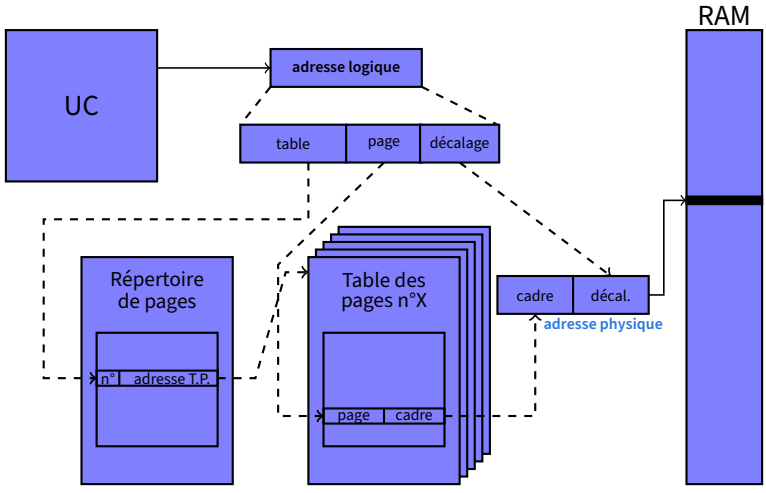
PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX

EXEMPLE

PAGINATION À DEUX NIVEAUX

EXEMPLE

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

PAGINATION À DEUX NIVEAUX

EXEMPLE

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

PAGINATION À DEUX NIVEAUX

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

PAGINATION À DEUX NIVEAUX

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 DEUX NIVEAUX

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* ($n_1 = 10, n_2 = 10, m = 12$)

PAGINATION À DEUX NIVEAUX

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 $\approx$ 4 Ko

PAGINATION À DEUX NIVEAUX

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 $\approx$ 4 Ko
  - 1 table de pages = 2^{10} lignes de 20 bits $\approx$ 2,5 Ko

PAGINATION À DEUX NIVEAUX

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 $\approx$ 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

CentraleSupélec
université PARIS-SACLAY

OUTLINE

Back to the begin - Back to the outline

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 $\geq$ 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 $\geq$ 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 $\geq$ 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 $\geq$ 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 $\geq$ 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 $\geq$ 1 Go
  - ✘ Somme des tailles des processus $\geq$ Capacité RAM
- Portions de code inutilisées
  - Traitement d'erreur $\longrightarrow$ rarement utilisé
  - Données (tableau, jeu, ...) $\longrightarrow$ pas tout en même temps
  - Bibliothèque $\longrightarrow$ 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 $\geq$ 1 Go
  - ✘ Somme des tailles des processus $\geq$ Capacité RAM
- Portions de code inutilisées
  - Traitement d'erreur $\longrightarrow$ rarement utilisé
  - Données (tableau, jeu, ...) $\longrightarrow$ pas tout en même temps
  - Bibliothèque $\longrightarrow$ 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 $\geq$ 1 Go
  - ✘ Somme des tailles des processus $\geq$ Capacité RAM
- Portions de code inutilisées
  - Traitement d'erreur $\longrightarrow$ rarement utilisé
  - Données (tableau, jeu, ...) $\longrightarrow$ pas tout en même temps
  - Bibliothèque $\longrightarrow$ très variable !

➠ ne charger que les pages utiles !

➠ laisser le reste sur le disque

CentraleSupélec
université PARIS-SACLAY

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) $\longrightarrow$ **mémoire virtuelle**

MÉMOIRE VIRTUELLE

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

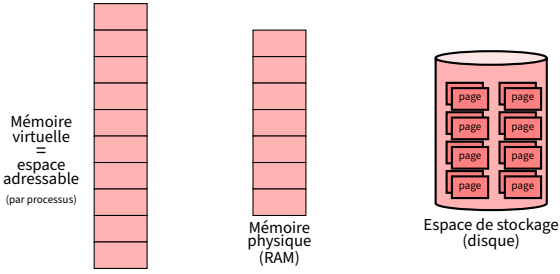


SCHÉMA DE PRINCIPE

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

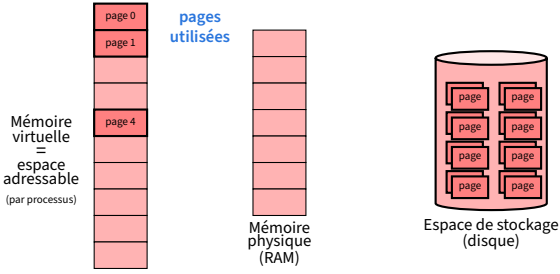


SCHÉMA DE PRINCIPE

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

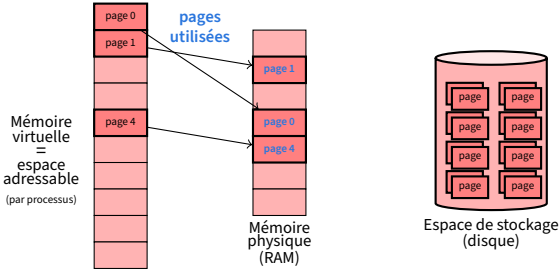


SCHÉMA DE PRINCIPE

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

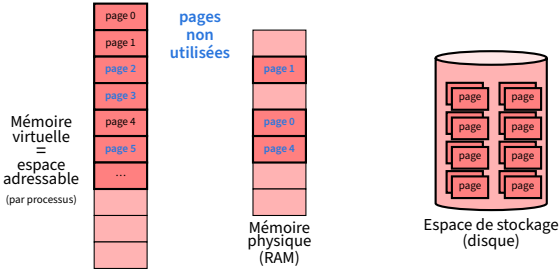


SCHÉMA DE PRINCIPE

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

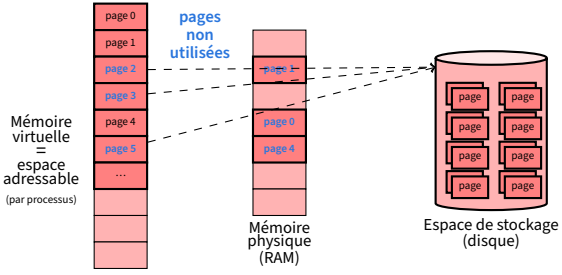


SCHÉMA DE PRINCIPE

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

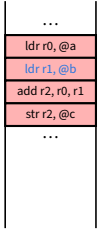
Chaque processus peut **adresser** plus d'espace
qu'il n'a effectivement en **mémoire physique**

- 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

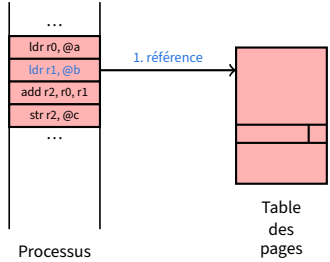
CentraleSupélec
université PARIS-SACLAY

PAGINATION À LA DEMANDE

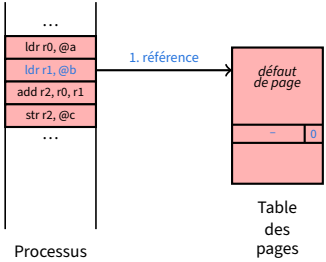


Processus

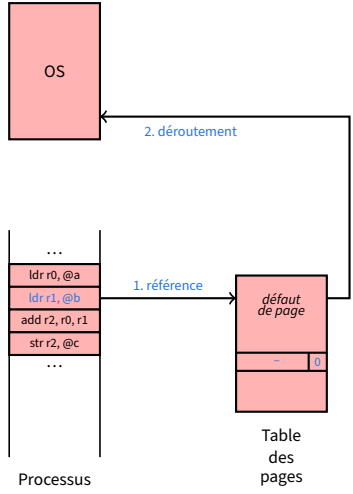
PAGINATION À LA DEMANDE



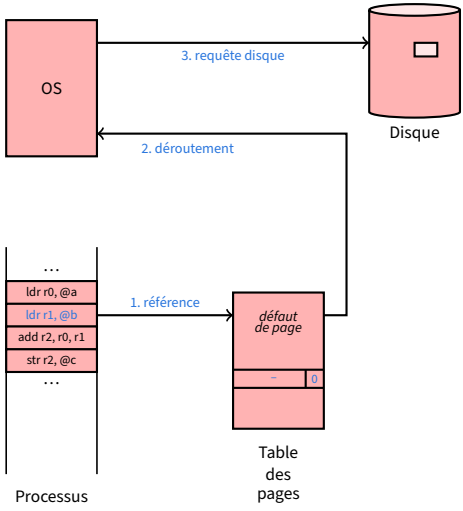
PAGINATION À LA DEMANDE



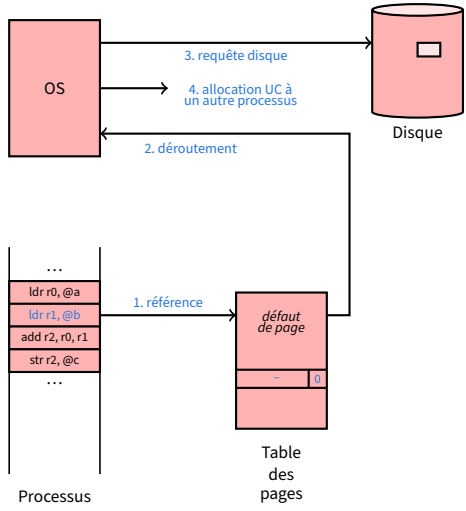
PAGINATION À LA DEMANDE



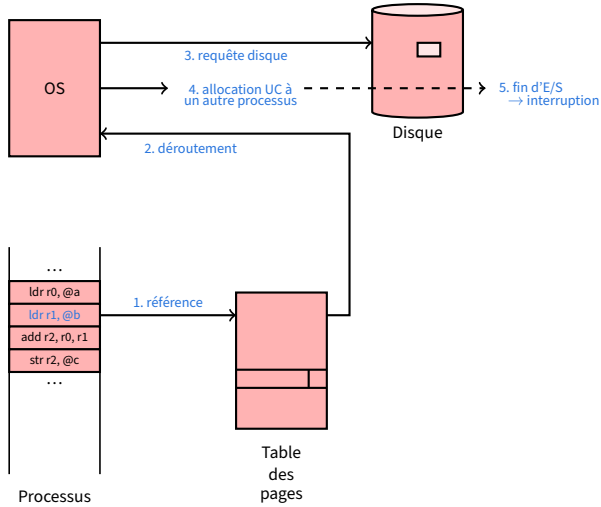
PAGINATION À LA DEMANDE



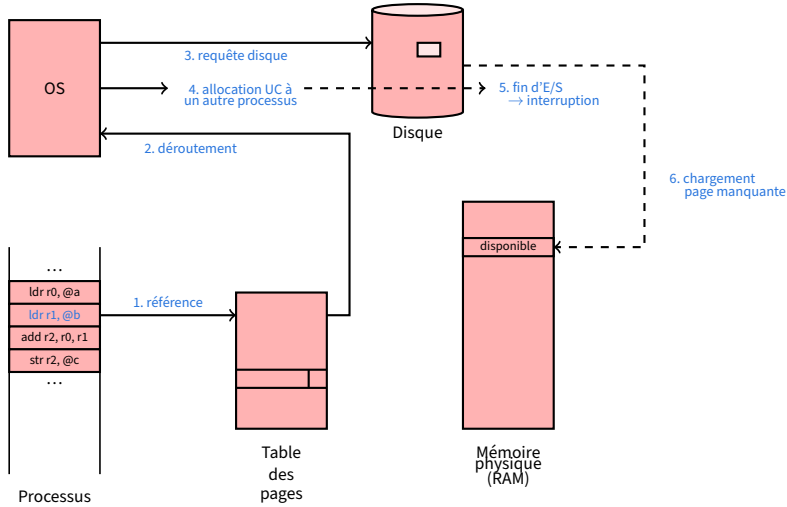
PAGINATION À LA DEMANDE



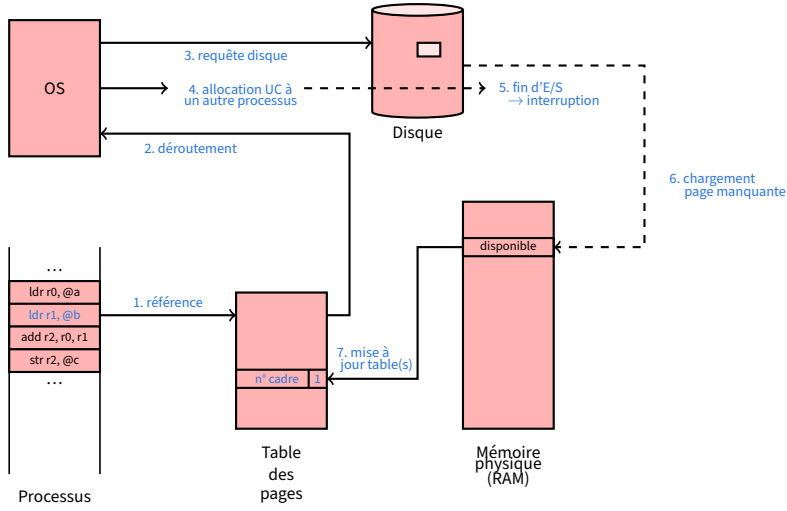
PAGINATION À LA DEMANDE



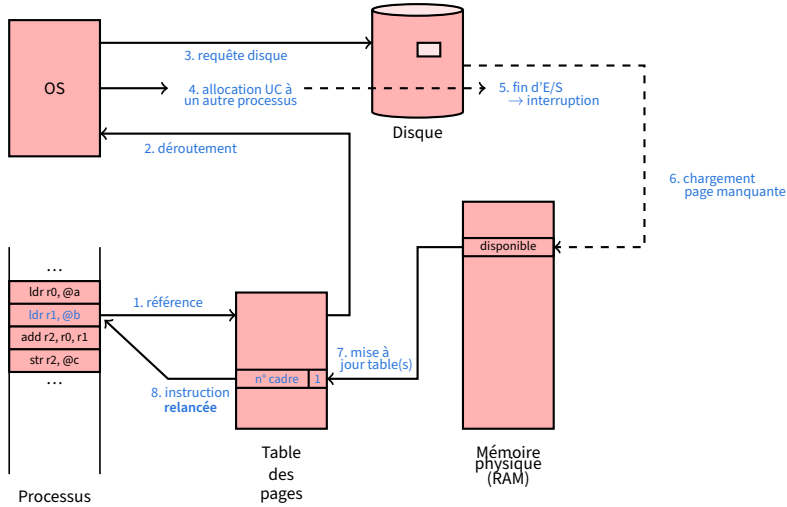
PAGINATION À LA DEMANDE



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
      $\rightarrow$ moins de processus $\rightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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
      $\longrightarrow$ moins de processus $\longrightarrow$ ralentissement
    - moins de cadre par processus
      $\longrightarrow$ plus de défauts $\longrightarrow$ 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 DES PAGES

ALLOCATION ÉQUITABLE

ALLOCATION DES PAGES

ALLOCATION ÉQUITABLE

- **Principe**

ALLOCATION DES PAGES
ALLOCATION ÉQUITABLE

- **Principe**
  - N cadres disponibles en RAM

ALLOCATION DES PAGES

ALLOCATION ÉQUITABLE

- **Principe**
  - N cadres disponibles en RAM
  - P processus

ALLOCATION DES PAGES

ALLOCATION ÉQUITABLE

- **Principe**
  - N cadres disponibles en RAM
  - P processus
  - Chaque processus reçoit $N \div P$ cadres

ALLOCATION DES PAGES
ALLOCATION ÉQUITABLE

- **Principe**
  - N cadres disponibles en RAM
  - P processus
  - Chaque processus reçoit $N \div P$ cadres
    - le reste sert de tampon

ALLOCATION DES PAGES
ALLOCATION ÉQUITABLE

- **Principe**
  - N cadres disponibles en RAM
  - P processus
  - Chaque processus reçoit $N \div P$ cadres
    - le reste sert de tampon

- **Inconvénient** $\longrightarrow$ tous les processus n'ont pas besoin de la même quantité de mémoire…

ALLOCATION DES PAGES

ALLOCATION PROPORTIONNELLE

- **Principe**
  - N cadres disponibles en RAM
  - P processus

ALLOCATION DES PAGES

ALLOCATION PROPORTIONNELLE

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

ALLOCATION DES 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 DES 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
  - P_i reçoit $(N - T) \times \frac{M_i}{\sum_i M_i}$ cadres

ALLOCATION DES 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
  - P_i reçoit $(N - T) \times \frac{M_i}{\sum_i M_i}$ cadres

- **Inconvénient** $\longrightarrow$ les petits processus font plus de défauts de pages

ALLOCATION DES PAGES

ALLOCATION BASÉE SUR LA PRIORITÉ

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

ALLOCATION DES 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 DES 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
  - P_i reçoit $(N - T) \times \frac{prio_i}{\sum_i prio_i}$

ALLOCATION DES 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
  - P_i reçoit $(N - T) \times \frac{prio_i}{\sum_i prio_i}$

- **Inconvénient** $\longrightarrow$ les processus moins prioritaires font plus de défauts de pages

REMPLACEMENT 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

REMPLACEMENT DE PAGES

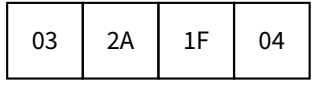
REMPLACEMENT DE PAGES

Exemple : 03 2A 1F 04

Défauts : 0

REMPLACEMENT DE PAGES

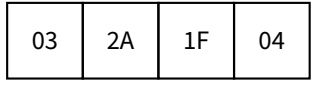
Exemple : 03 2A 1F 04



Défauts : 4

REMPLACEMENT DE PAGES

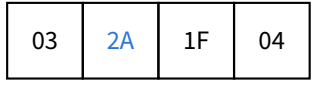
Exemple : 03 2A 1F 04 2A



Défauts : 4

REMPLACEMENT DE PAGES

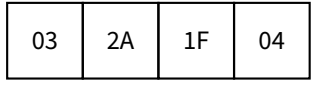
Exemple : 03 2A 1F 04 2A



Défauts : 4

REMPLACEMENT DE PAGES

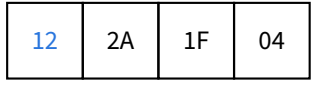
Exemple : 03 2A 1F 04 2A 12



Défauts : 4

REMPLACEMENT DE PAGES

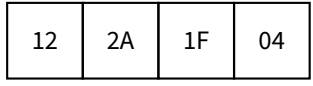
Exemple : 03 2A 1F 04 2A 12



Défauts : 5

REMPLACEMENT DE PAGES

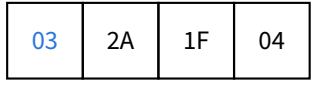
Exemple : 03 2A 1F 04 2A 12 03



Défauts : 5

REMPLACEMENT DE PAGES

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



Défauts : 6

REPRÉSENTATION VISUELLE

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

Exemple ...

REPRÉSENTATION VISUELLE

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

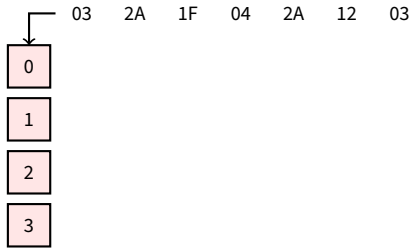
03 2A 1F 04 2A 12 03

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

CentraleSupélec
université PARIS-SACLAY

REPRÉSENTATION VISUELLE

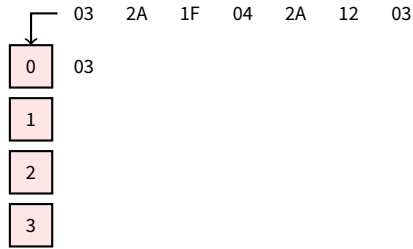
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une ligne par cadre de page alloué au processus

REPRÉSENTATION VISUELLE

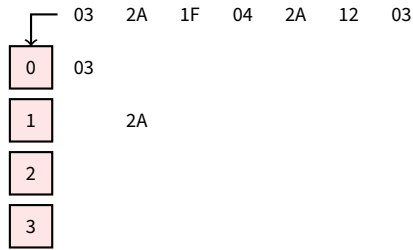
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

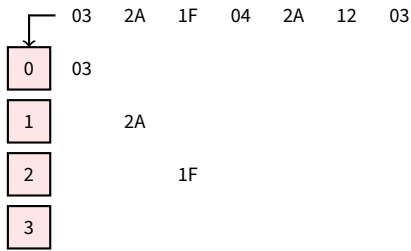
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

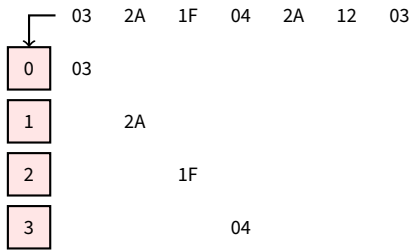
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

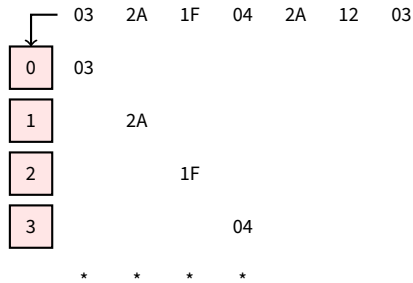
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Une page par colonne

REPRÉSENTATION VISUELLE

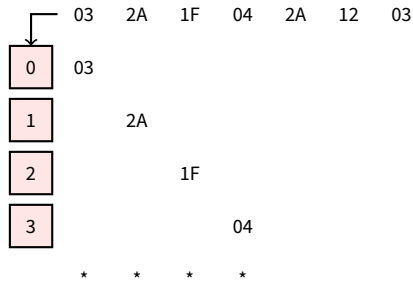
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Marquer les défauts au fur et à mesure

REPRÉSENTATION VISUELLE

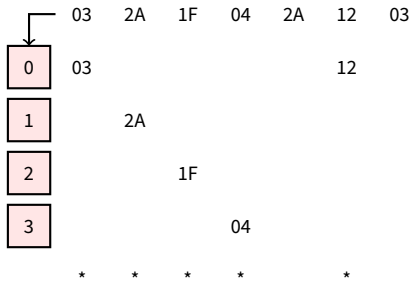
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Pas de défaut → laisser en blanc

REPRÉSENTATION VISUELLE

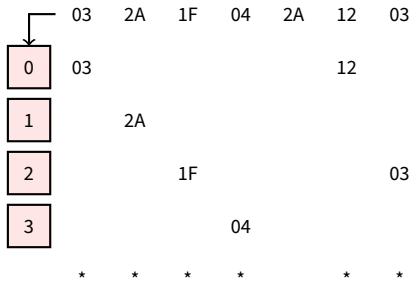
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Et on continue…

REPRÉSENTATION VISUELLE

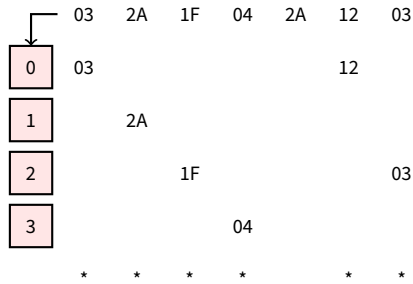
Figurer l'état du cache dans le temps
pour compter le nombre de défauts



Et on continue…

REPRÉSENTATION VISUELLE

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



Total : 6 défauts

Et on continue…

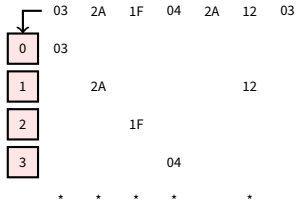
SOLUTION OPTIMALE

Il existe une politique de remplacement optimale…

…si on connaît à l'avance les références aux pages !

SOLUTION OPTIMALE

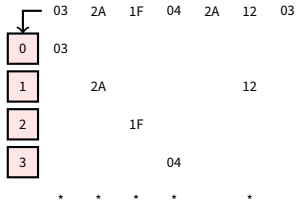
Il existe une politique de remplacement optimale…
…si on connaît à l'avance les références aux pages !



Total : 5 défauts

SOLUTION OPTIMALE

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



Total : 5 défauts

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

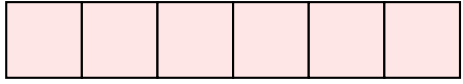
Retirer la page la plus ancienne

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

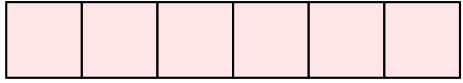


ALGORITHMES TYPE FIFO

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

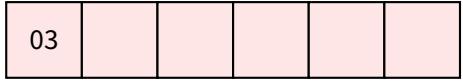


ALGORITHMES TYPE FIFO

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

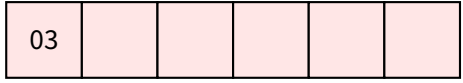


ALGORITHMES TYPE FIFO

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

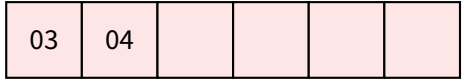


ALGORITHMES TYPE FIFO

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

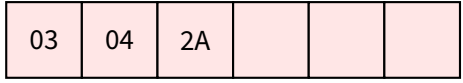


ALGORITHMES TYPE FIFO

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

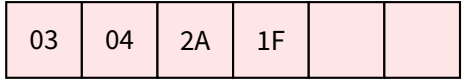


ALGORITHMES TYPE FIFO

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

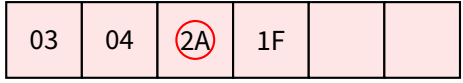


ALGORITHMES TYPE FIFO

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

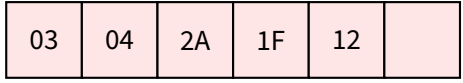


ALGORITHMES TYPE FIFO

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

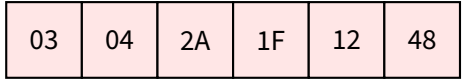


ALGORITHMES TYPE FIFO

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

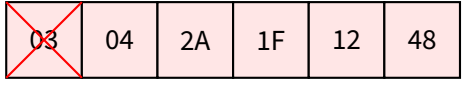


ALGORITHMES TYPE FIFO

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

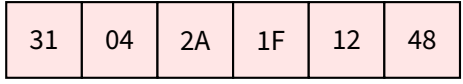


ALGORITHMES TYPE FIFO

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

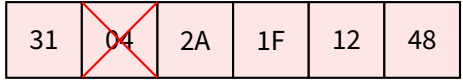


ALGORITHMES TYPE FIFO

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

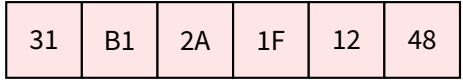


ALGORITHMES TYPE FIFO

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

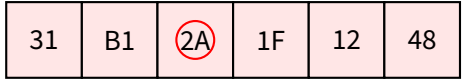


ALGORITHMES TYPE FIFO

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

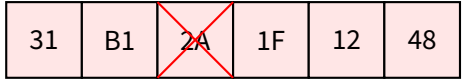


ALGORITHMES TYPE FIFO

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

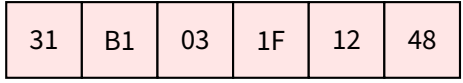


ALGORITHMES TYPE FIFO

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

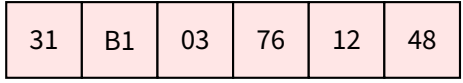


ALGORITHMES TYPE FIFO

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

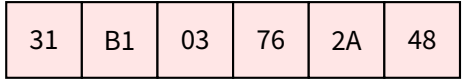


ALGORITHMES TYPE FIFO

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

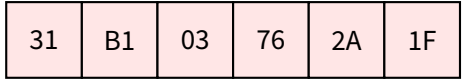


ALGORITHMES TYPE FIFO

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

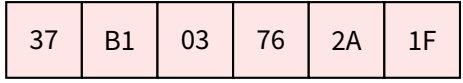


ALGORITHMES TYPE FIFO

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

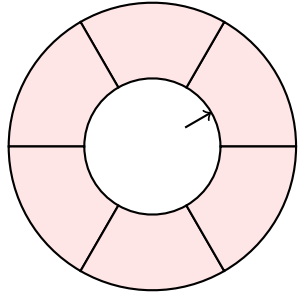


ALGORITHMES TYPE FIFO

Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

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

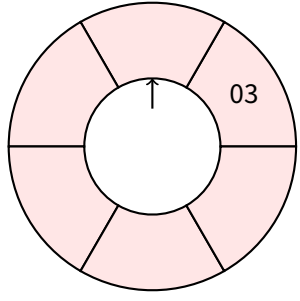


ALGORITHMES TYPE FIFO

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

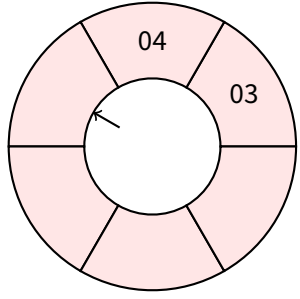


ALGORITHMES TYPE FIFO

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

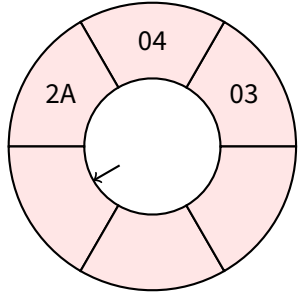


ALGORITHMES TYPE FIFO

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

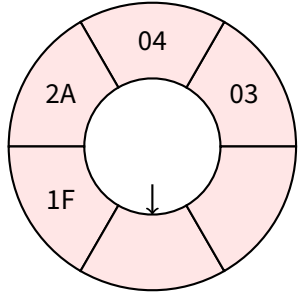


ALGORITHMES TYPE FIFO

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

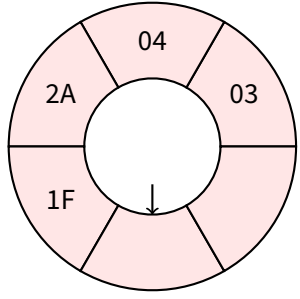


ALGORITHMES TYPE FIFO

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

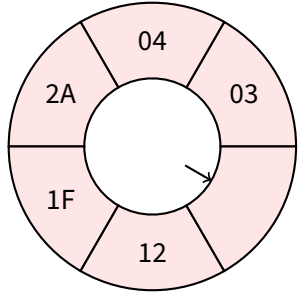


ALGORITHMES TYPE FIFO

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

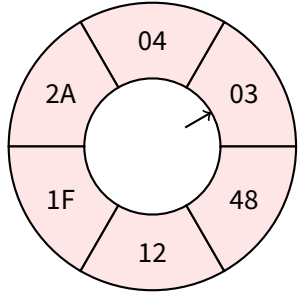


ALGORITHMES TYPE FIFO

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

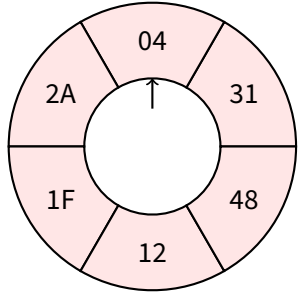


ALGORITHMES TYPE FIFO

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

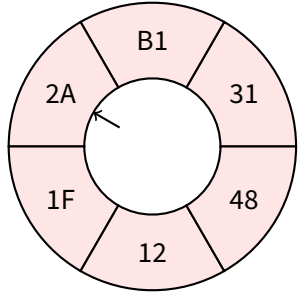


ALGORITHMES TYPE FIFO

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

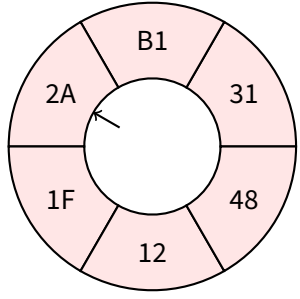


ALGORITHMES TYPE FIFO

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

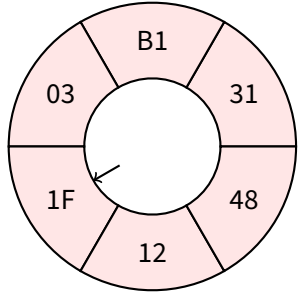


ALGORITHMES TYPE FIFO

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

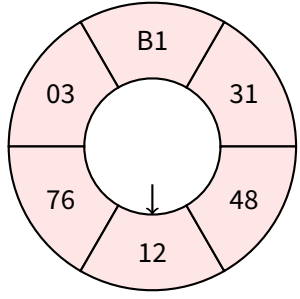


ALGORITHMES TYPE FIFO

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

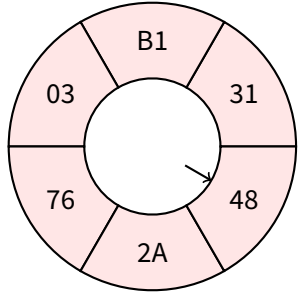


ALGORITHMES TYPE FIFO

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

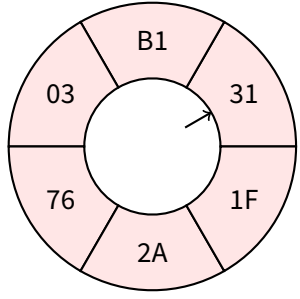


ALGORITHMES TYPE FIFO

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

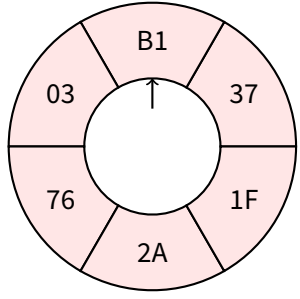


ALGORITHMES TYPE FIFO

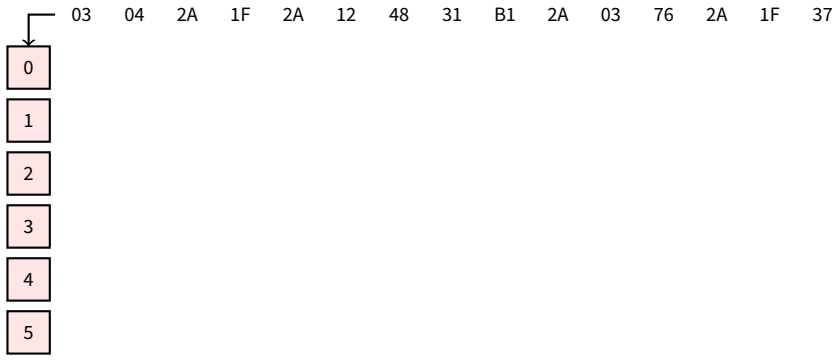
Retirer la page la plus ancienne

Une autre façon d'implémenter l'algorithme

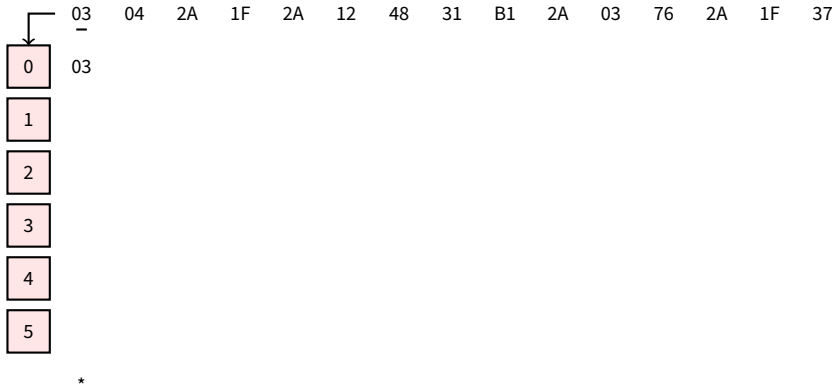
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



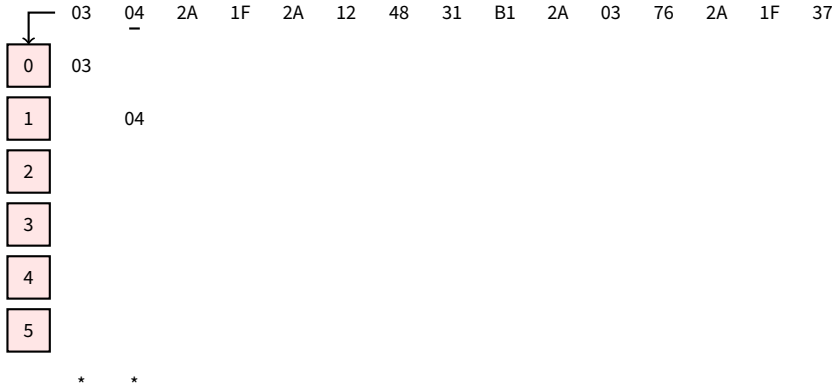
COMPTER LES DÉFAUTS



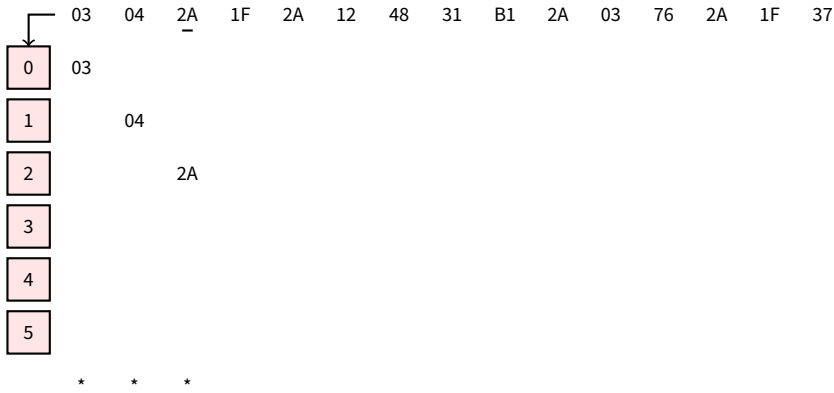
COMPTER LES DÉFAUTS



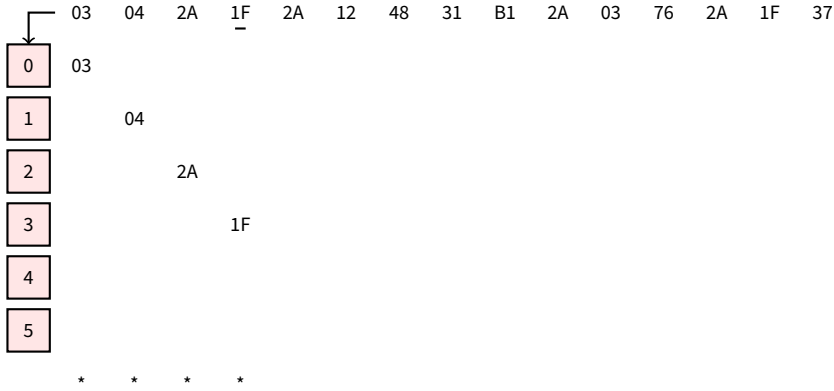
COMPTER LES DÉFAUTS



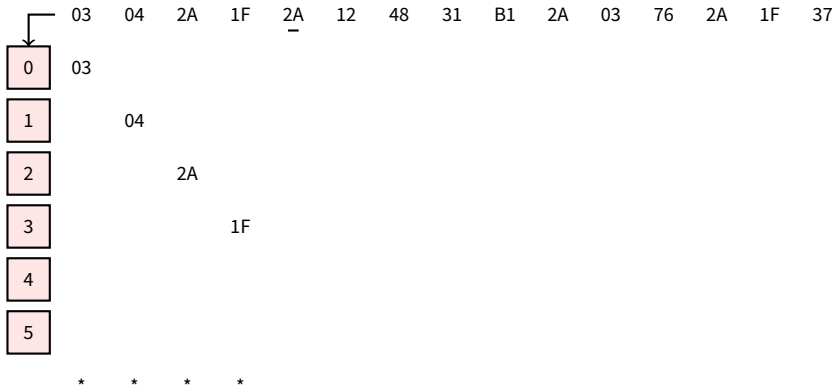
COMPTER LES DÉFAUTS



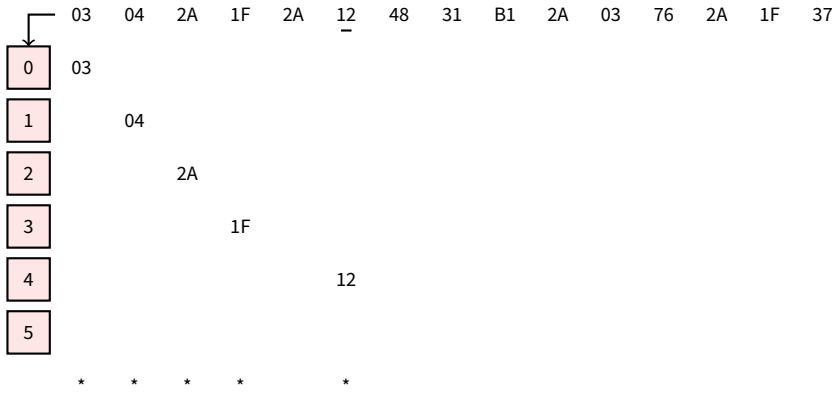
COMPTER LES DÉFAUTS



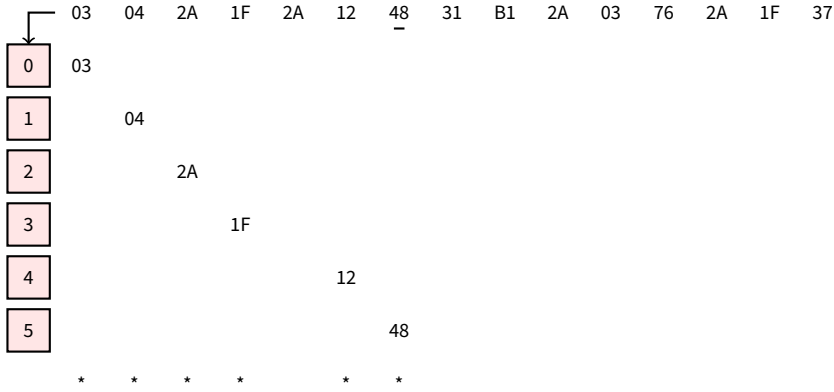
COMPTER LES DÉFAUTS



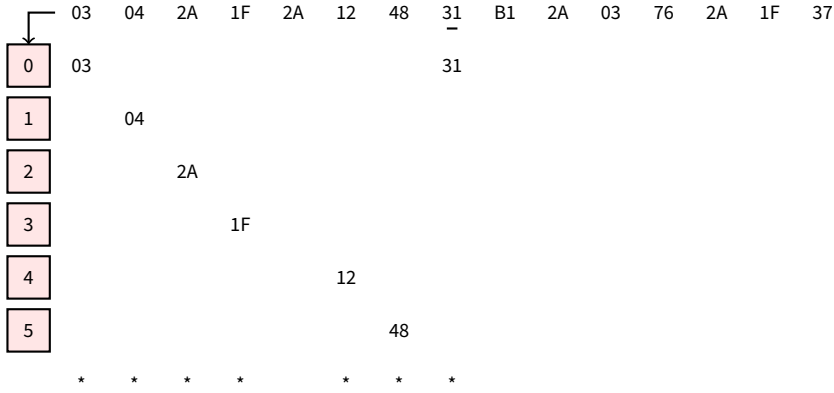
COMPTER LES DÉFAUTS



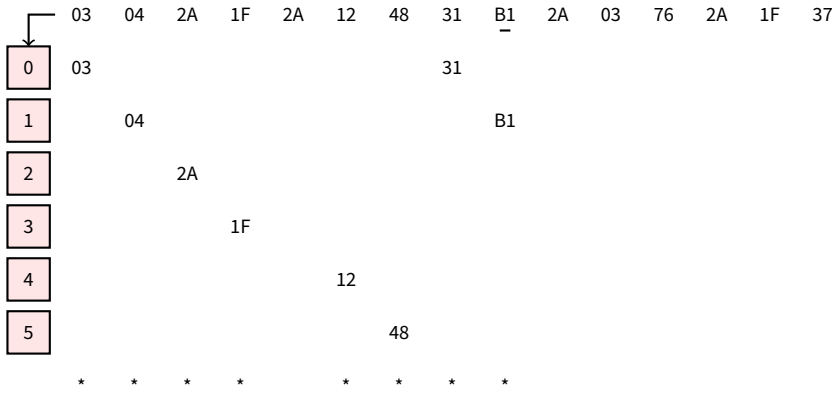
COMPTER LES DÉFAUTS



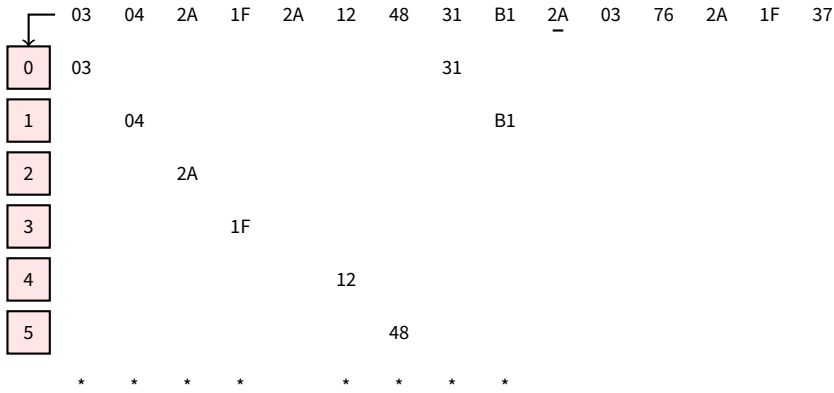
COMPTER LES DÉFAUTS



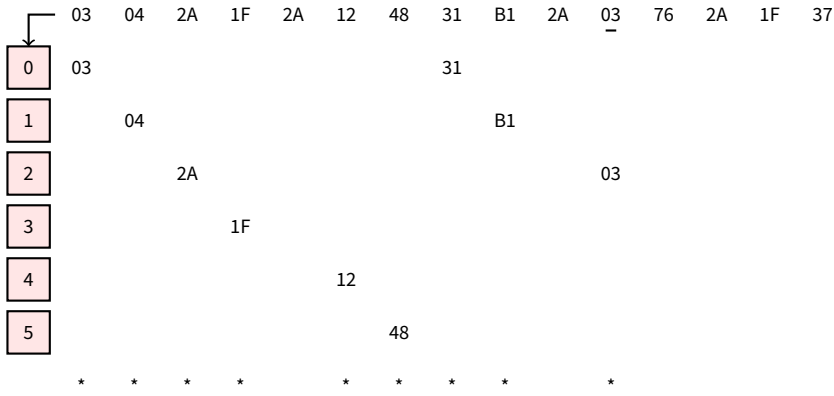
COMPTER LES DÉFAUTS



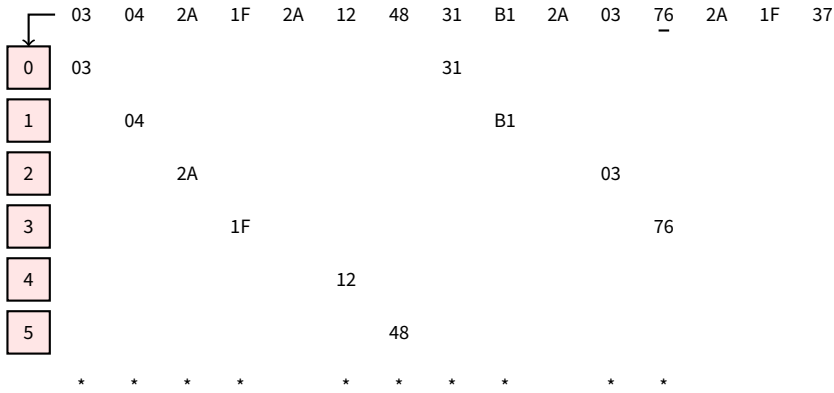
COMPTER LES DÉFAUTS



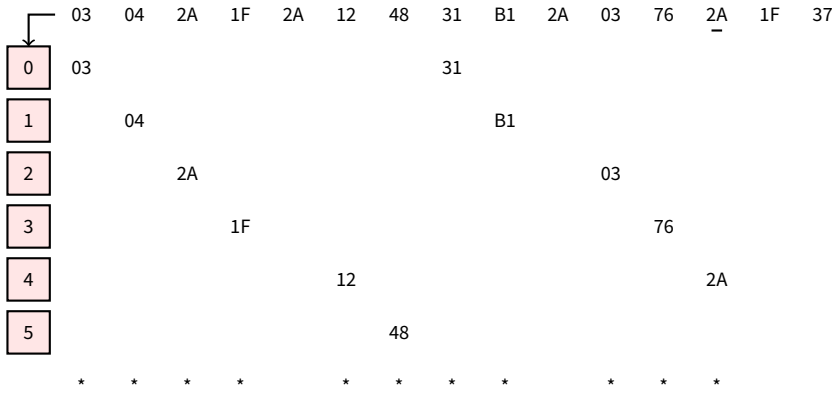
COMPTER LES DÉFAUTS



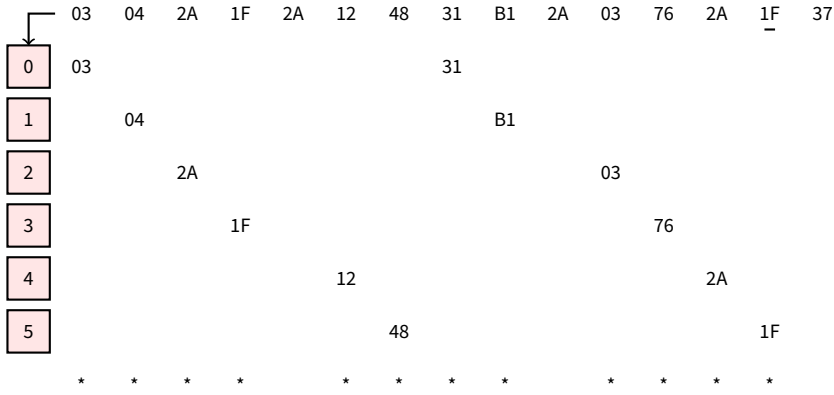
COMPTER LES DÉFAUTS



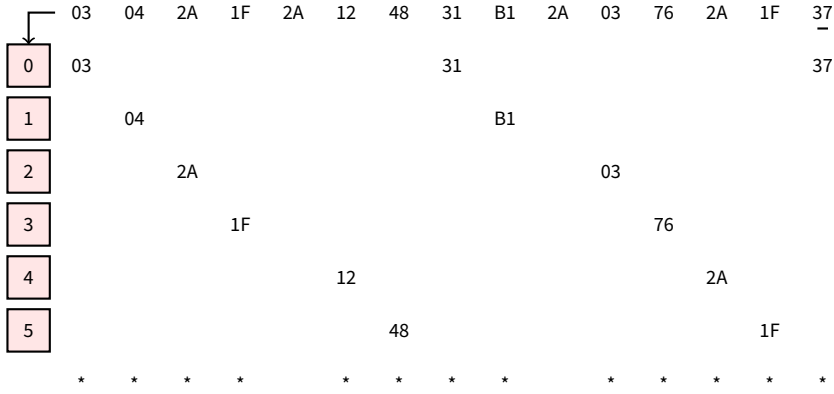
COMPTER LES DÉFAUTS



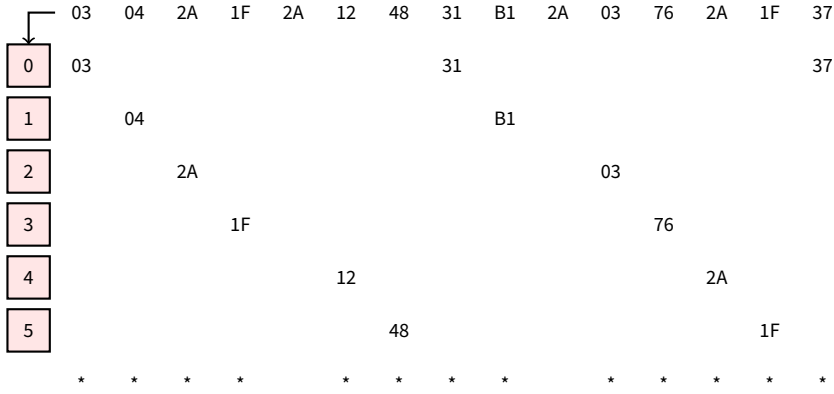
COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS



COMPTER LES DÉFAUTS



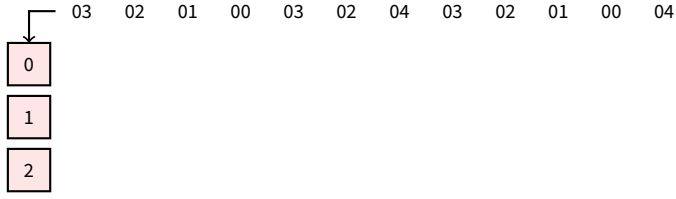
Total : 13 défauts

ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

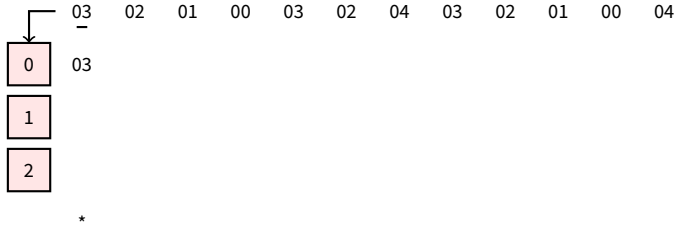


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

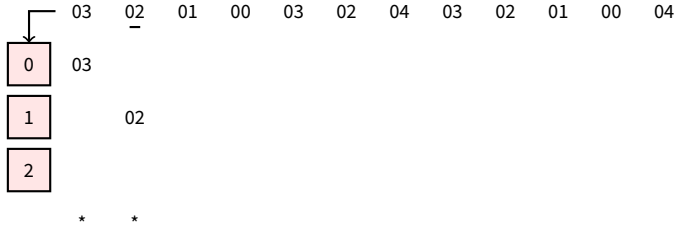


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

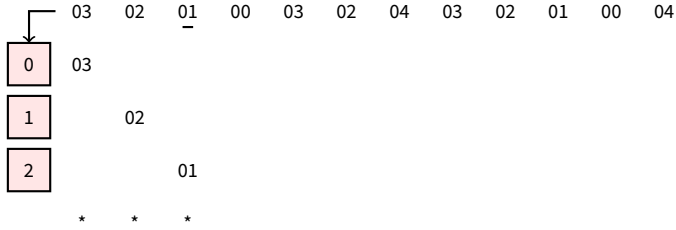


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

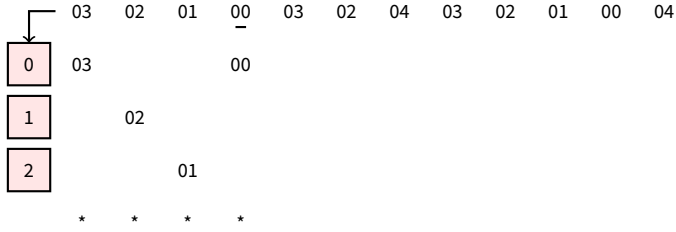


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

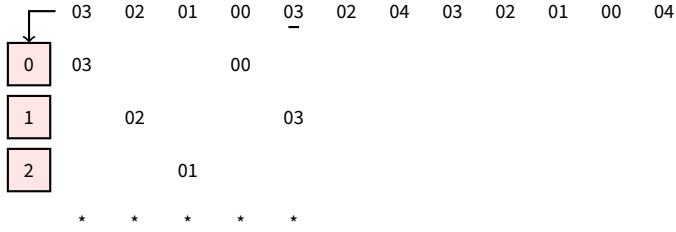


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

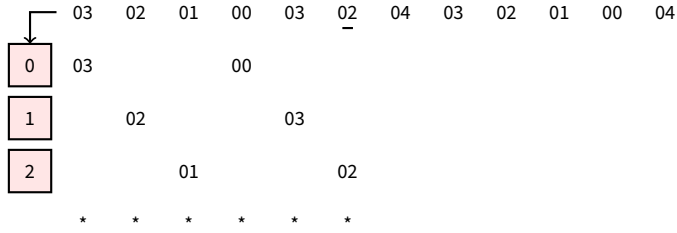


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

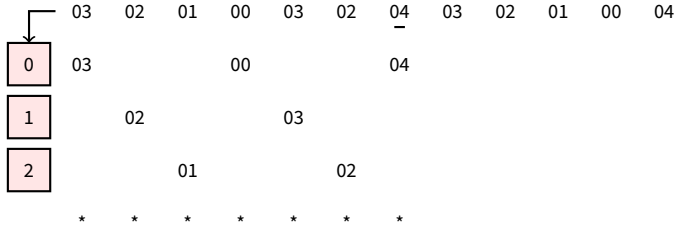


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

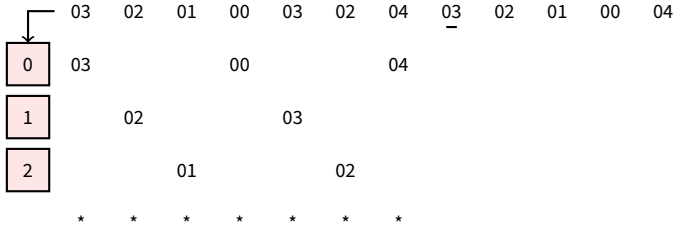


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

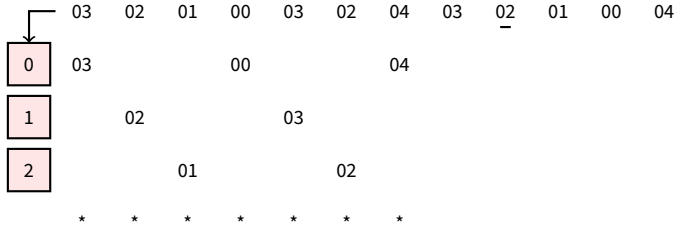


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

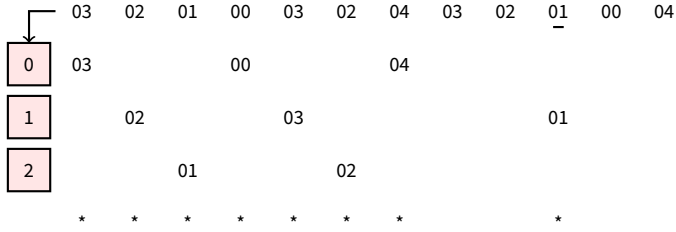


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

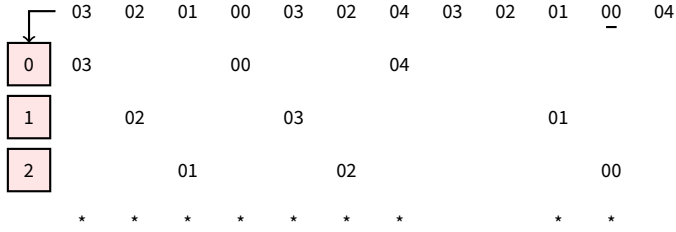


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

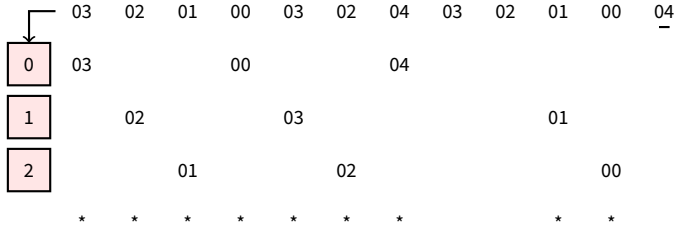


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres

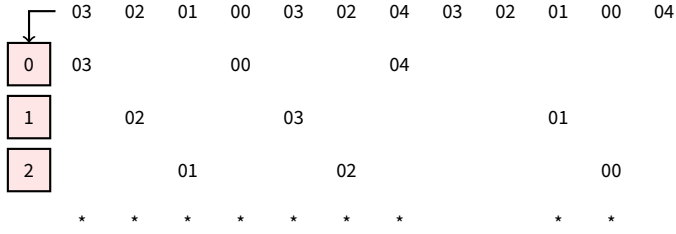


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 3 cadres



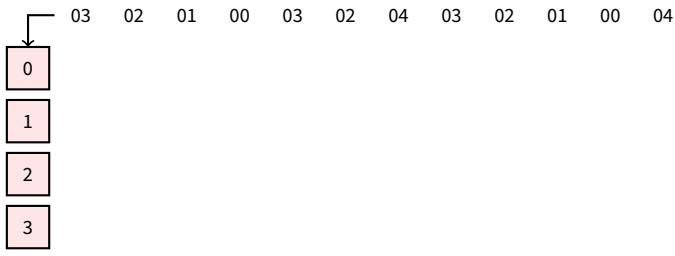
Total : 9 défauts

ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

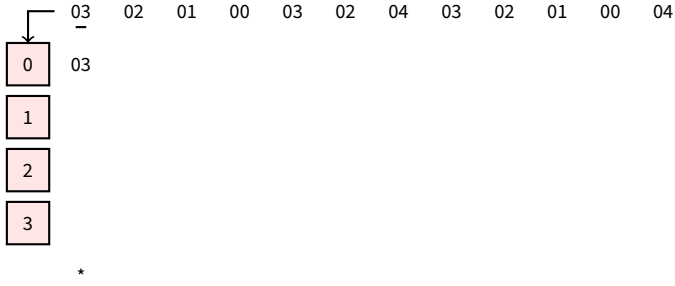


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

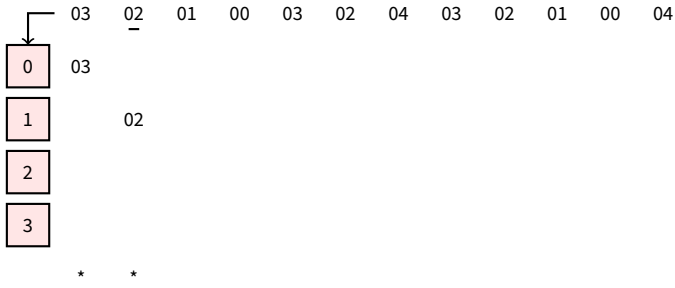


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

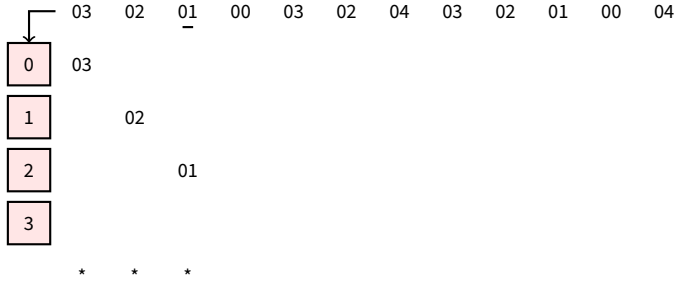


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

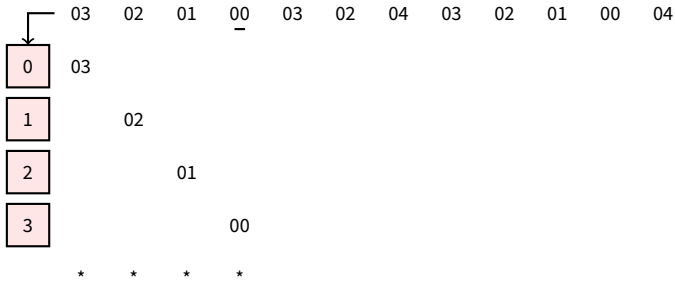


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

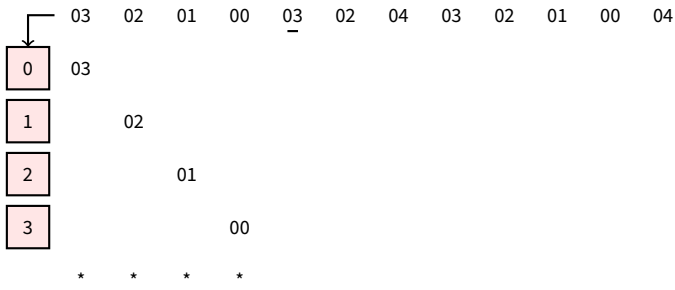


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

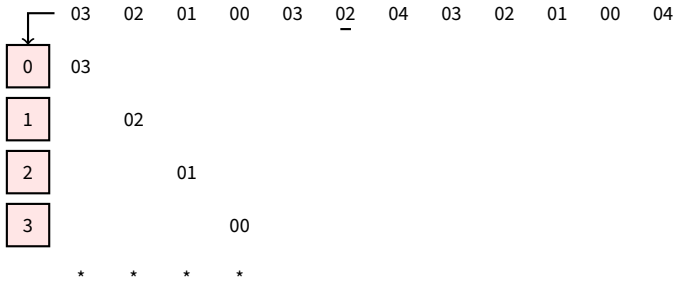


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

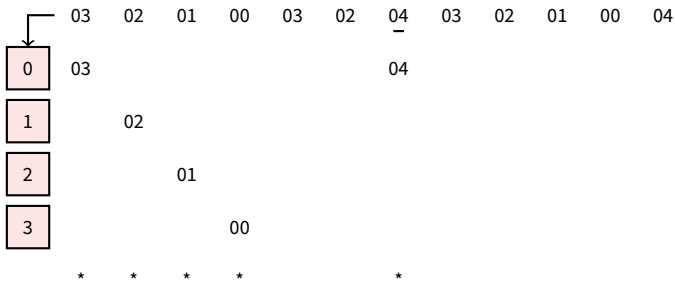


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

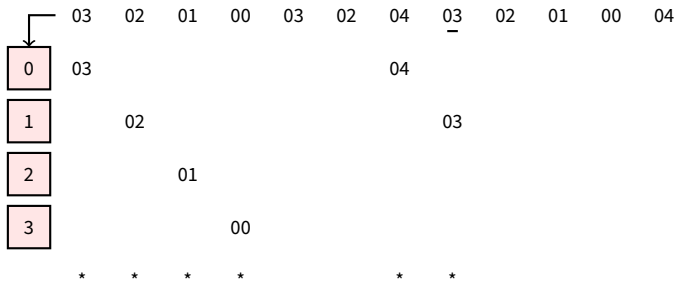


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

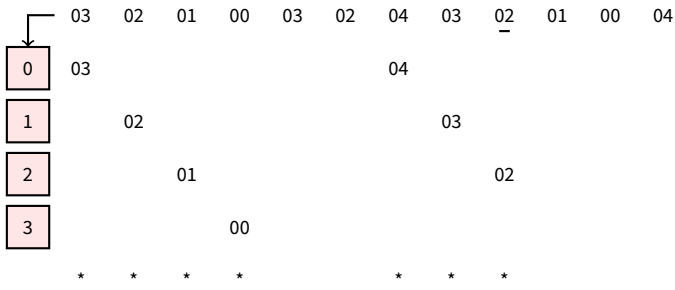


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

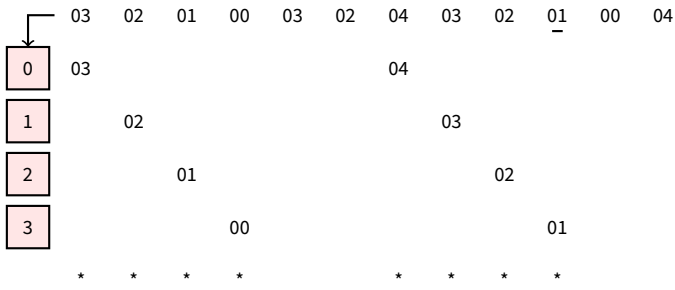


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

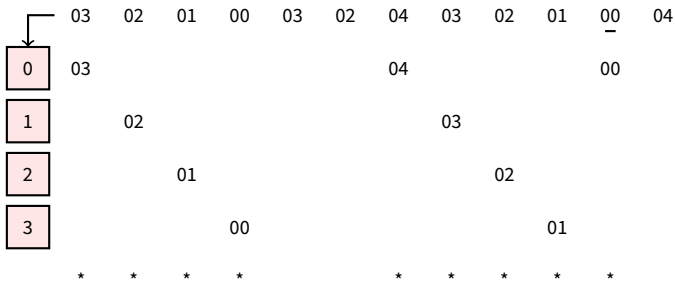


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

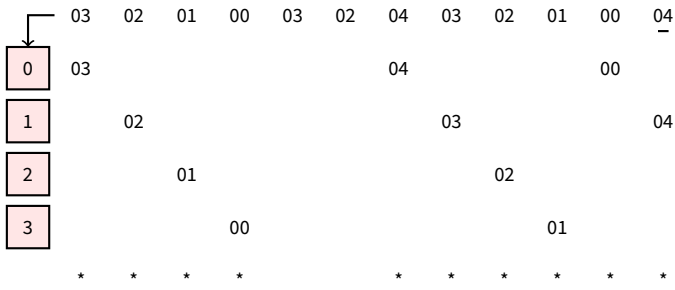


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

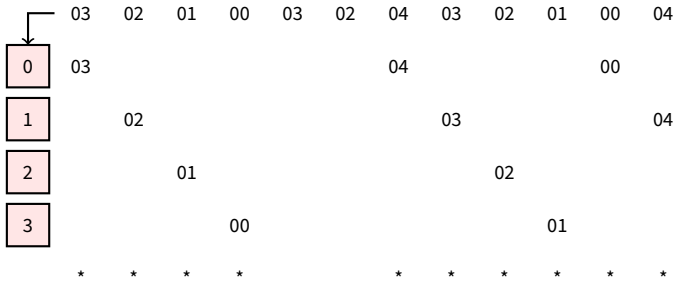


ANOMALIE DE BELADY

Sur certaines instances…

plus de cadres $\longrightarrow$ plus de défauts !

Exemple : avec 4 cadres

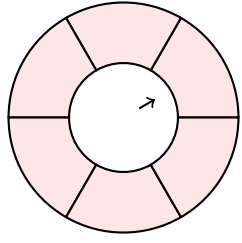


Total : 10 défauts

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)

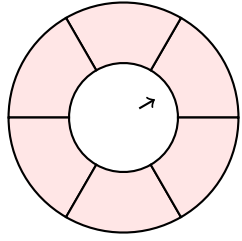
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

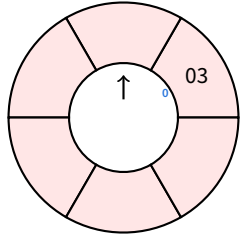
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

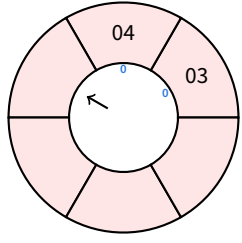
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

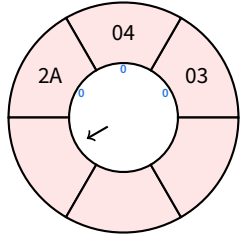
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

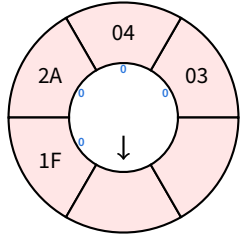
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

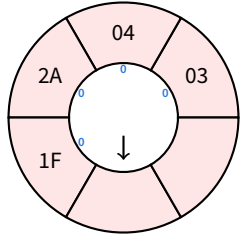
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

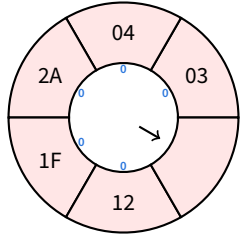
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

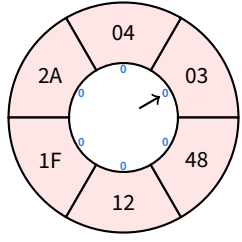
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

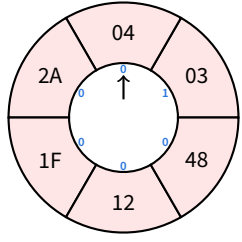
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

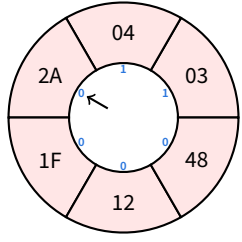
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

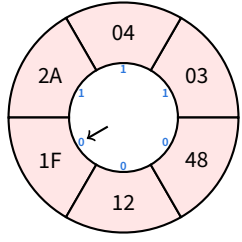
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

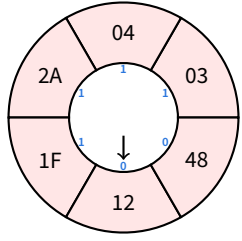
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

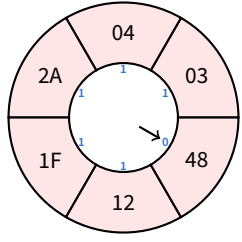
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

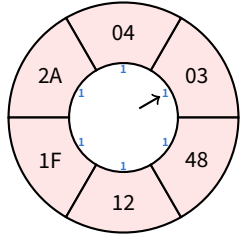
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

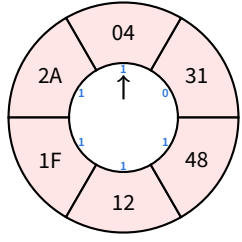
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

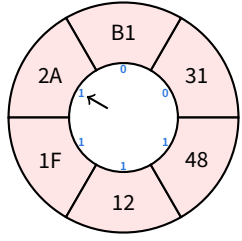
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

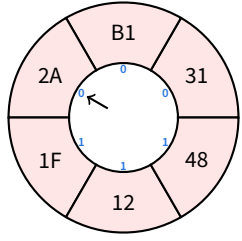
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

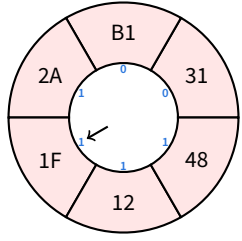
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

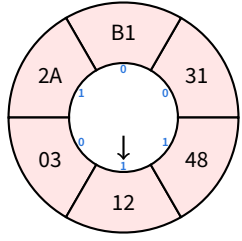
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

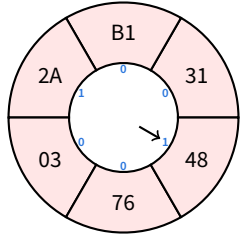
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

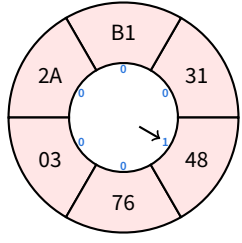
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

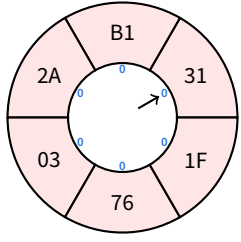
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

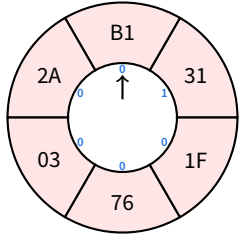
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

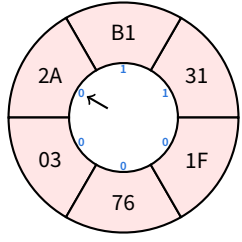
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

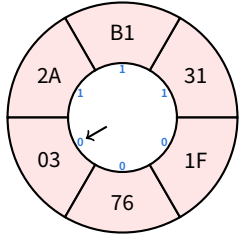
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

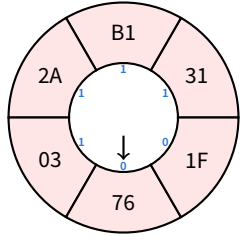
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

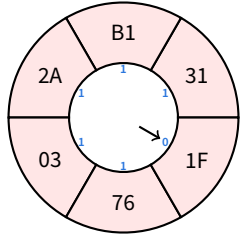
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

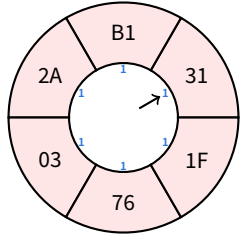
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



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)

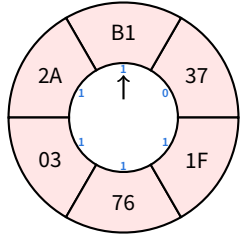
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



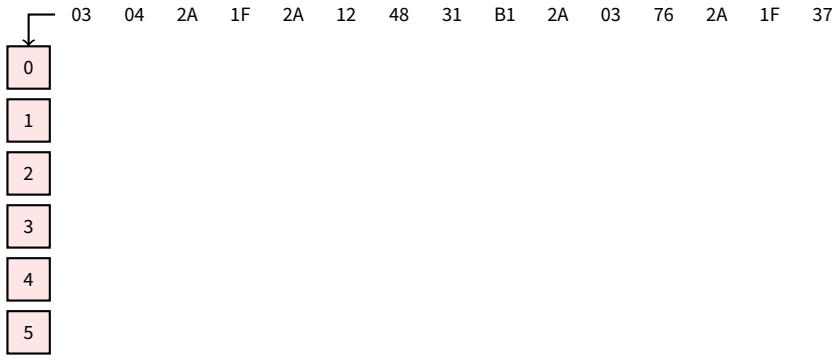
FIFO - SECONDE CHANCE

- **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)

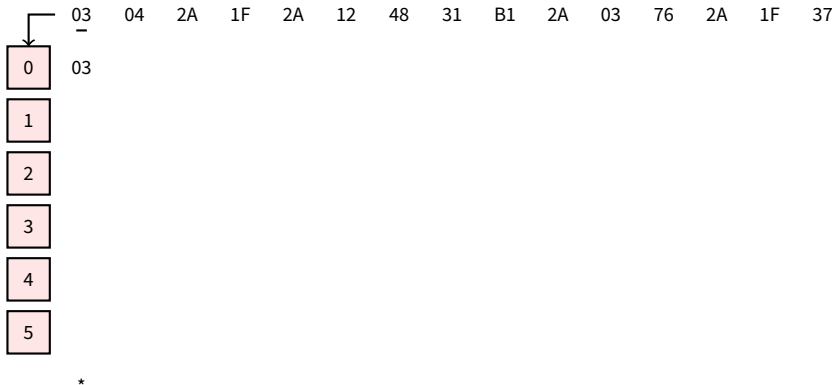
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



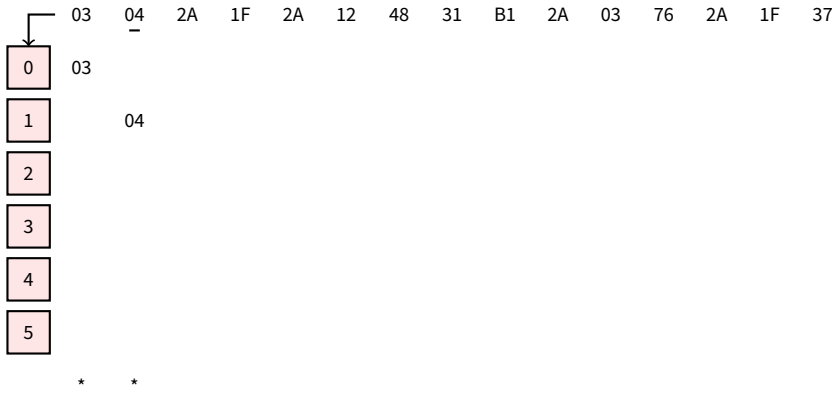
FIFO - SECONDE CHANCE



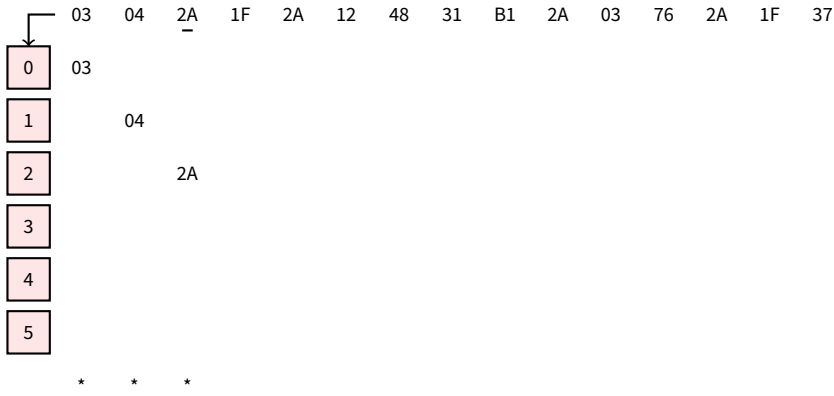
FIFO - SECONDE CHANCE



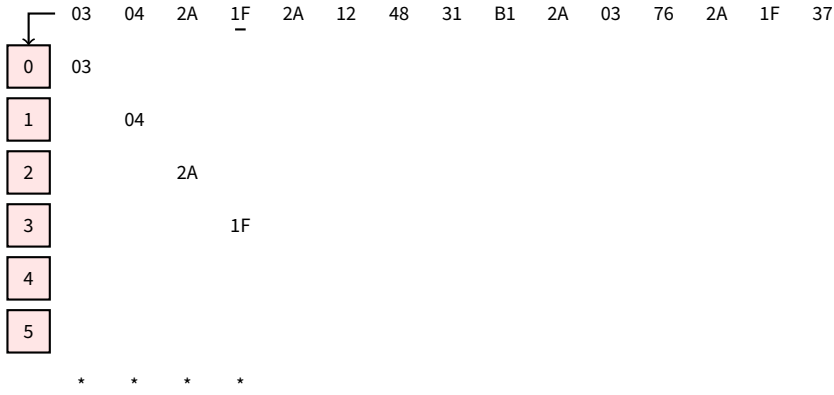
FIFO - SECONDE CHANCE



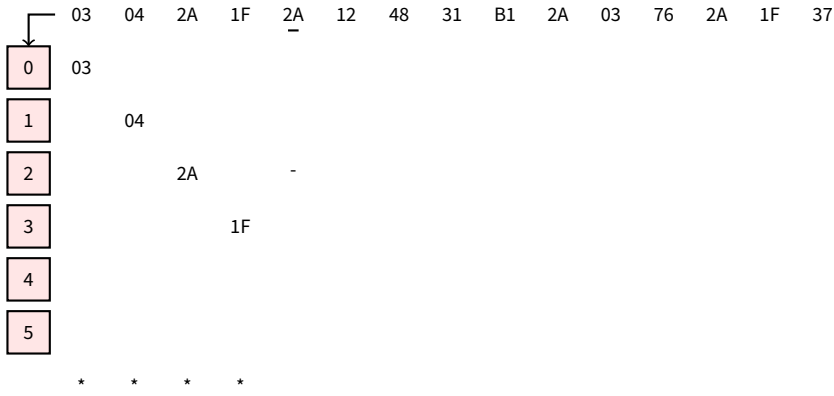
FIFO - SECONDE CHANCE



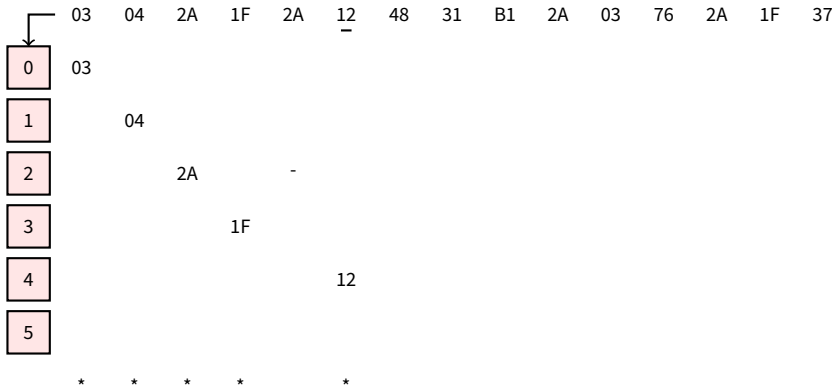
FIFO - SECONDE CHANCE



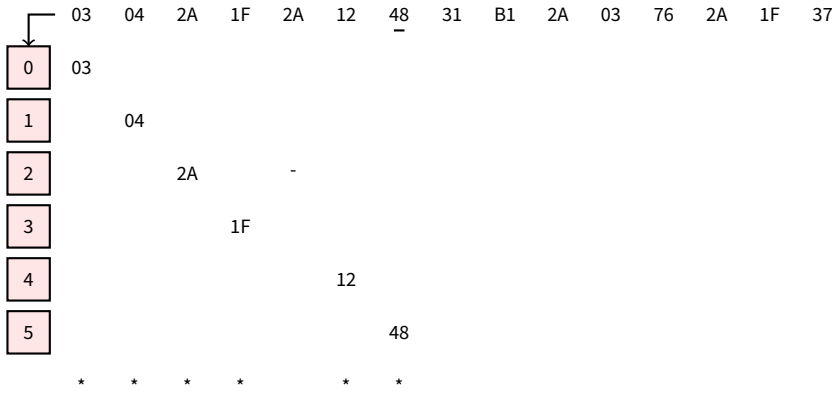
FIFO - SECONDE CHANCE



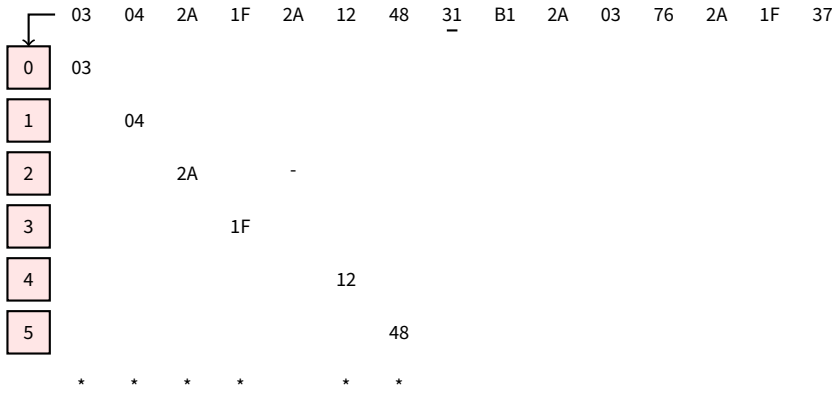
FIFO - SECONDE CHANCE



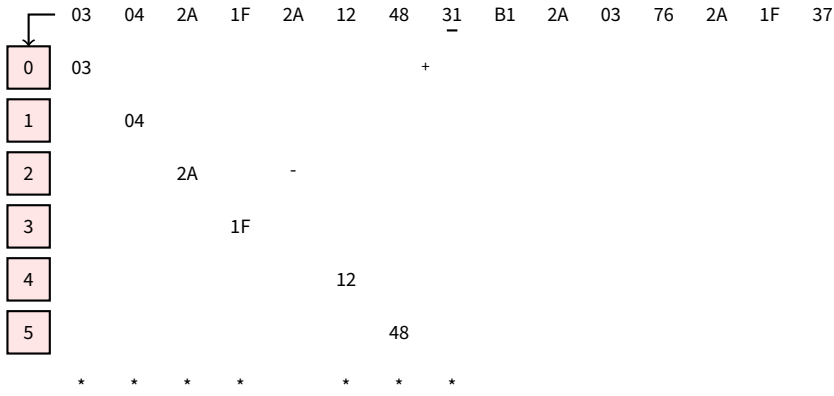
FIFO - SECONDE CHANCE



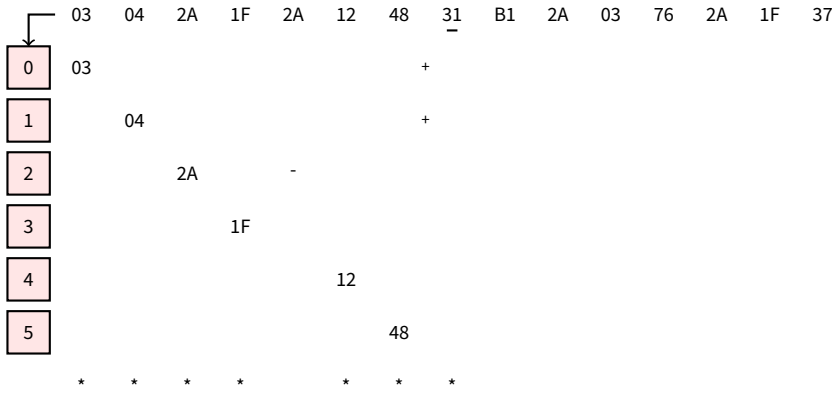
FIFO - SECONDE CHANCE



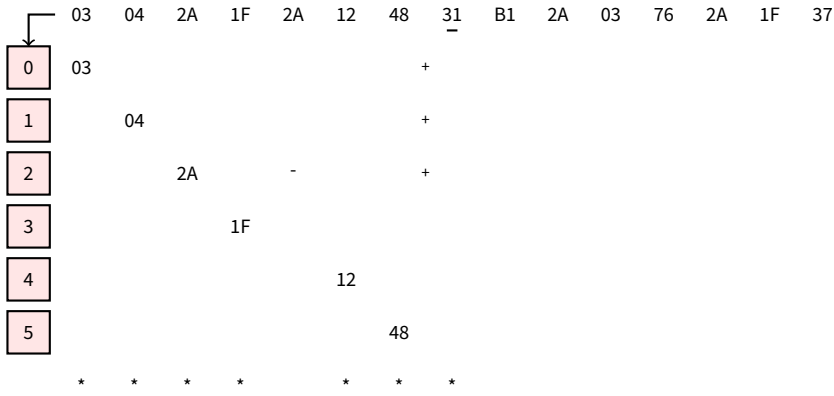
FIFO - SECONDE CHANCE



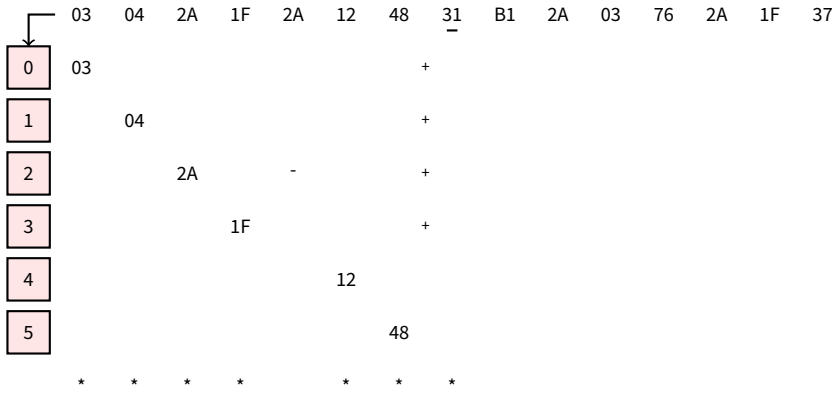
FIFO - SECONDE CHANCE



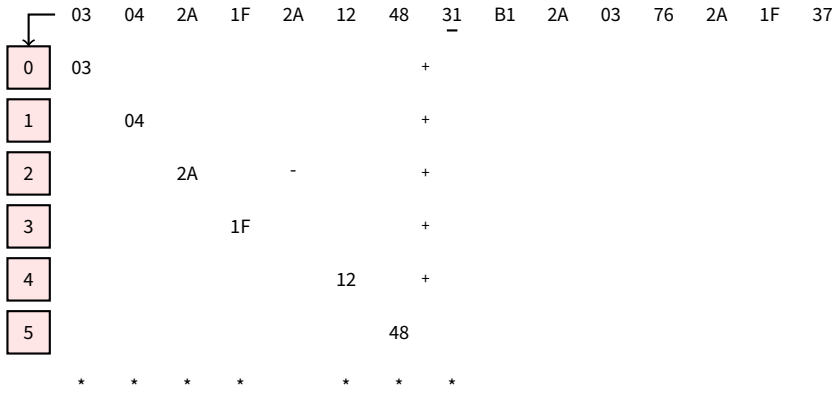
FIFO - SECONDE CHANCE



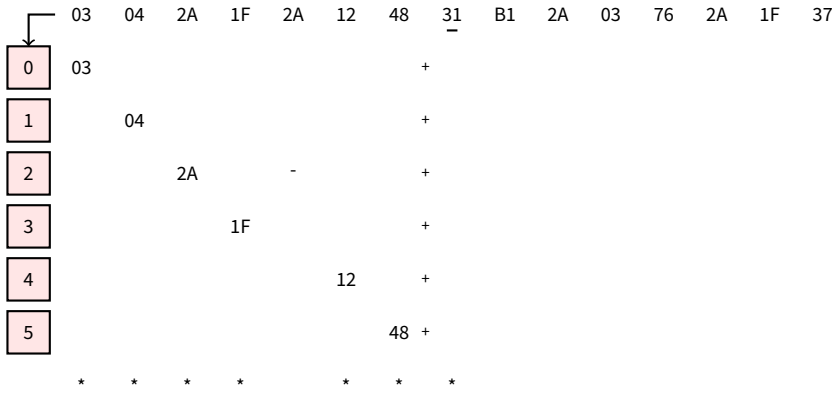
FIFO - SECONDE CHANCE



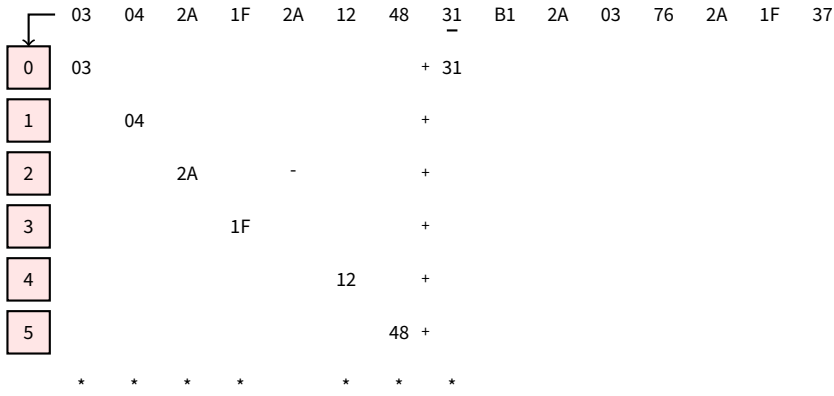
FIFO - SECONDE CHANCE



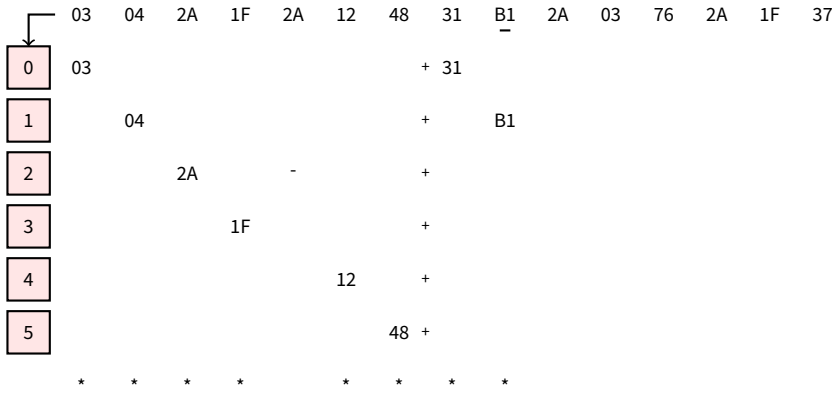
FIFO - SECONDE CHANCE



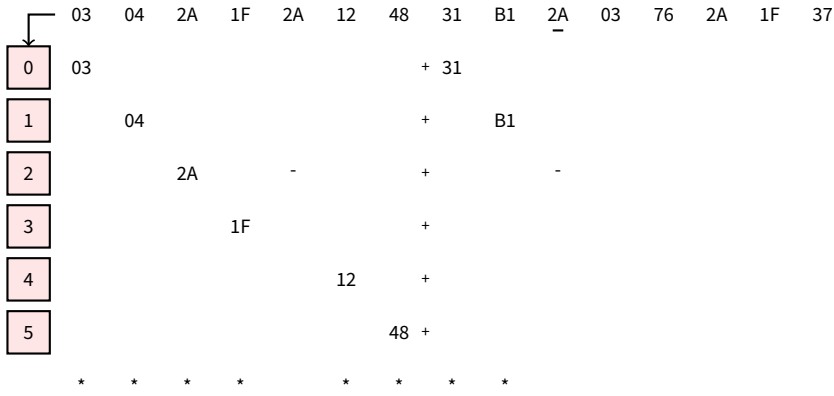
FIFO - SECONDE CHANCE



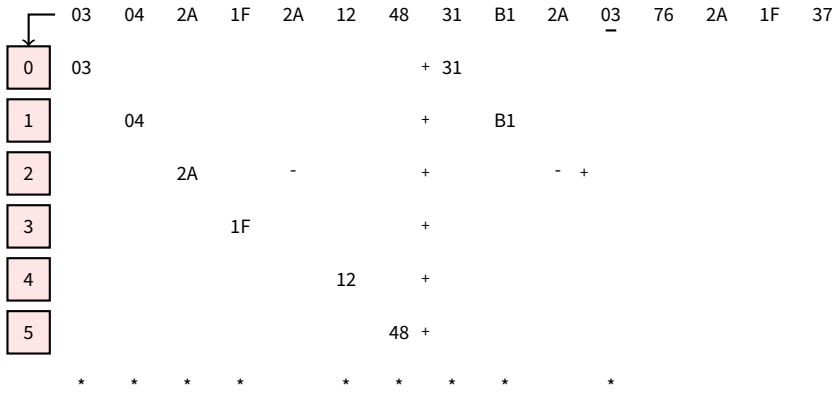
FIFO - SECONDE CHANCE



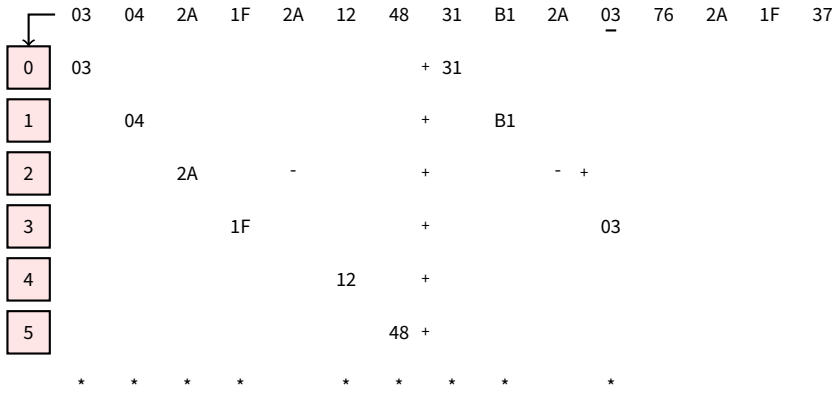
FIFO - SECONDE CHANCE



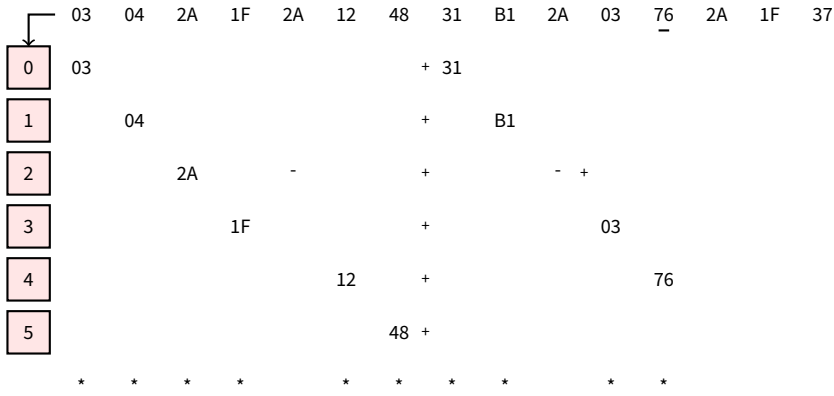
FIFO - SECONDE CHANCE



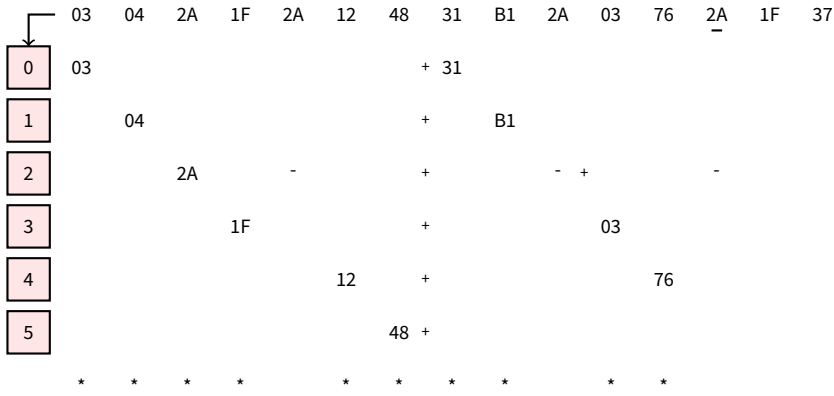
FIFO - SECONDE CHANCE



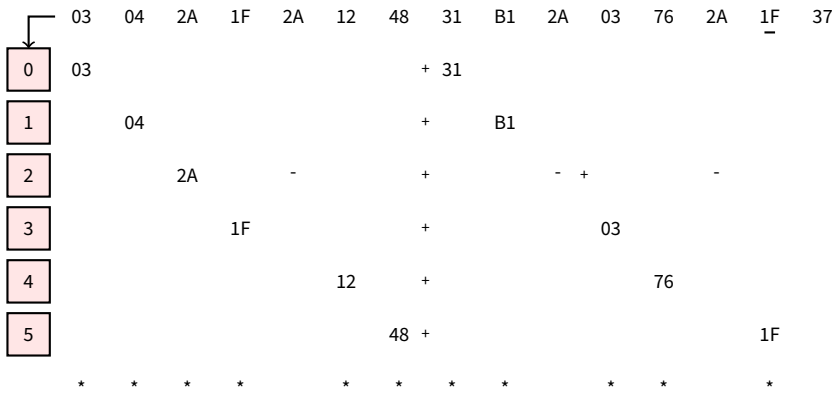
FIFO - SECONDE CHANCE



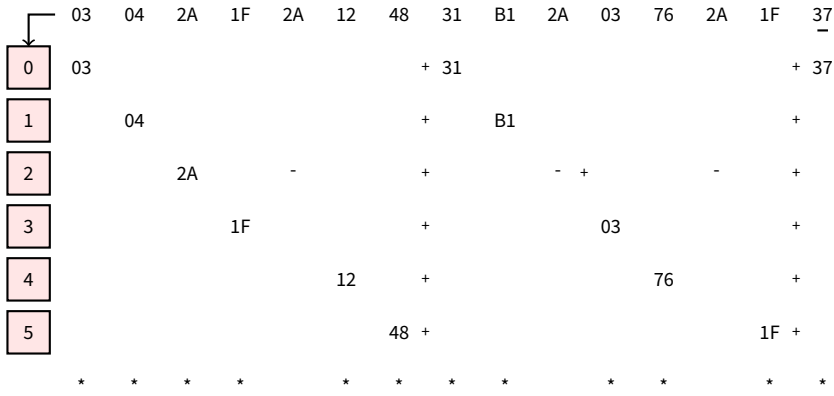
FIFO - SECONDE CHANCE



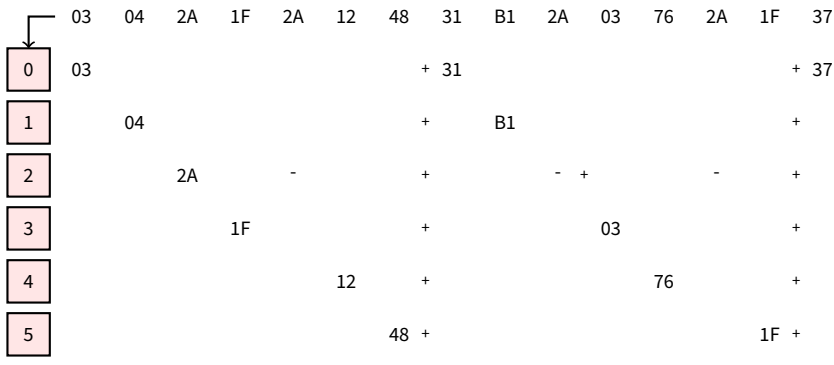
FIFO - SECONDE CHANCE



FIFO - SECONDE CHANCE



FIFO - SECONDE CHANCE



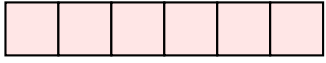
Total : 12 défauts

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 $\longrightarrow$ FIFO

- **Exemple**

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



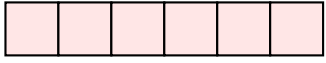
Page	Usage	Date
03	0	
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



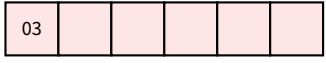
Page	Usage	Date
03	0	
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



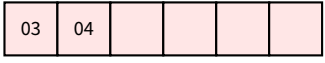
Page	Usage	Date
03	1	1
04	0	
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



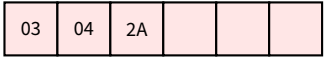
Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	0	
2A	0	
31	0	
37	0	
48	0	
76	0	
B1	0	

LEAST FREQUENTLY USED - LFU

- **Principe**
  - Noter le taux d'utilisation de chaque page du processus.
    - choisir la page la moins utilisée
    - en cas d'égalité $\longrightarrow$ FIFO

- **Exemple**

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



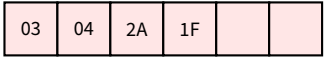
Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	0	
2A	1	3
31	0	
37	0	
48	0	
76	0	
B1	0	

LEAST FREQUENTLY USED - LFU

- **Principe**
  - Noter le taux d'utilisation de chaque page du processus.
    - choisir la page la moins utilisée
    - en cas d'égalité $\longrightarrow$ FIFO

- **Exemple**

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



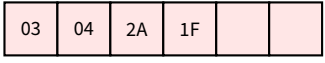
Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	1	4
2A	1	3
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



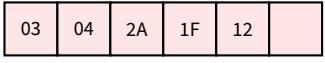
Page	Usage	Date
03	1	1
04	1	2
12	0	
1F	1	4
2A	2	3
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



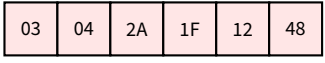
Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	0	
37	0	
48	0	
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



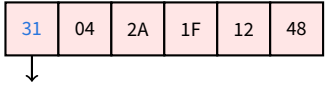
Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	0	
37	0	
48	1	7
76	0	
B1	0	

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 $\longrightarrow$ FIFO

- **Exemple**

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



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

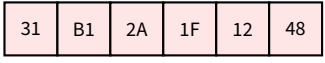
Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	2	3
31	1	8
37	0	
48	1	7
76	0	
B1	0	

LEAST FREQUENTLY USED - LFU

- **Principe**
  - Noter le taux d'utilisation de chaque page du processus.
    - choisir la page la moins utilisée
    - en cas d'égalité $\longrightarrow$ FIFO

- **Exemple**

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



Page	Usage	Date
03	1	1
04	1	2
12	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 $\longrightarrow$ FIFO

- **Exemple**

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

31	B1	2A	1F	12	48

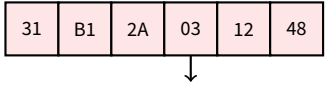
Page	Usage	Date
03	1	1
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	0	
B1	1	9

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 $\longrightarrow$ 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})

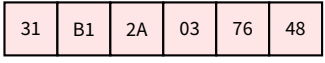
Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	0	
B1	1	9

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 $\longrightarrow$ FIFO

- **Exemple**

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



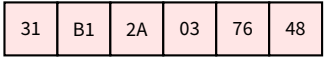
Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	3	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

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 $\longrightarrow$ FIFO

- **Exemple**

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



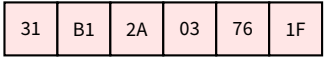
Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	1	4
2A	4	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

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 $\longrightarrow$ FIFO

- **Exemple**

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



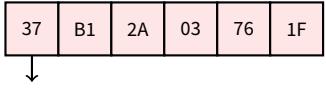
Page	Usage	Date
03	2	11
04	1	2
12	1	6
1F	2	14
2A	4	3
31	1	8
37	0	
48	1	7
76	1	12
B1	1	9

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 $\longrightarrow$ FIFO

- **Exemple**

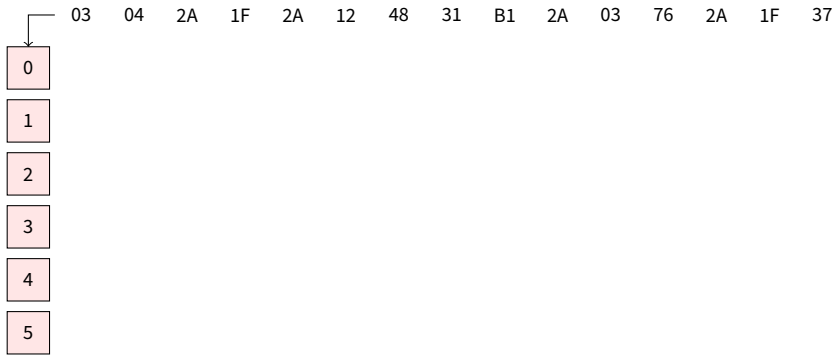
03 04 2A 1F 2A 12 48 31 B1 2A 03 76 2A 1F 37



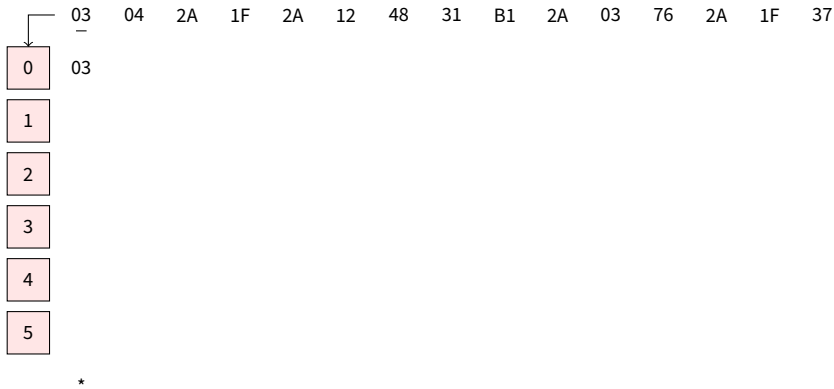
Usage = 1, Date = min({8,9,12})

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

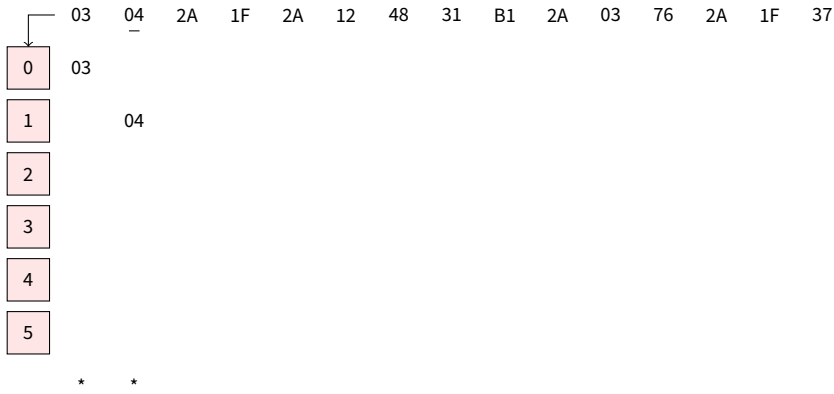
LEAST FREQUENTLY USED - LFU



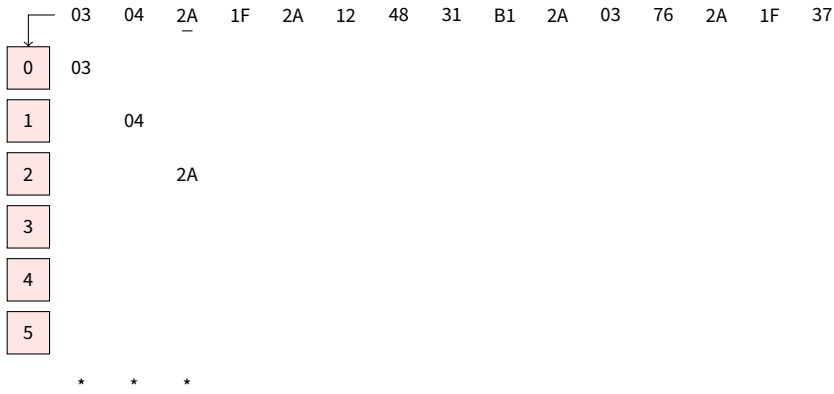
LEAST FREQUENTLY USED - LFU



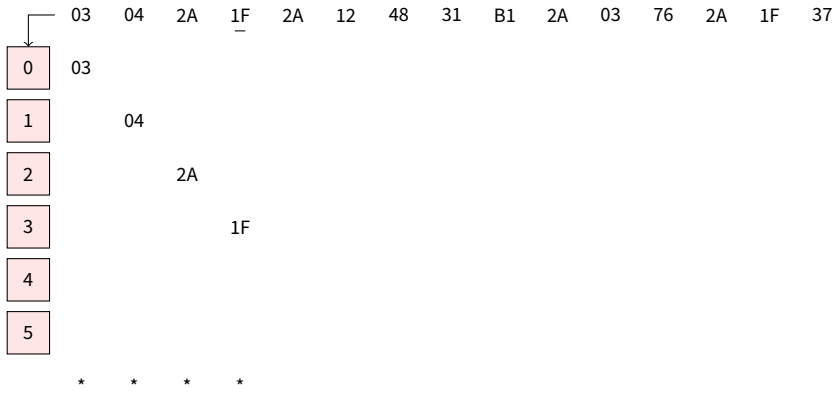
LEAST FREQUENTLY USED - LFU



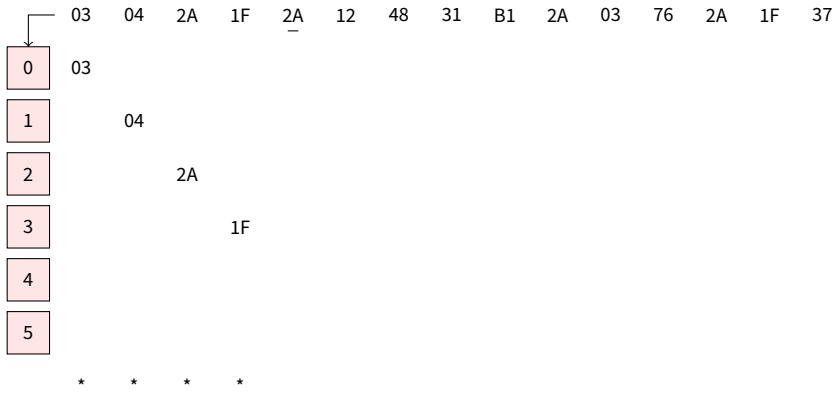
LEAST FREQUENTLY USED - LFU



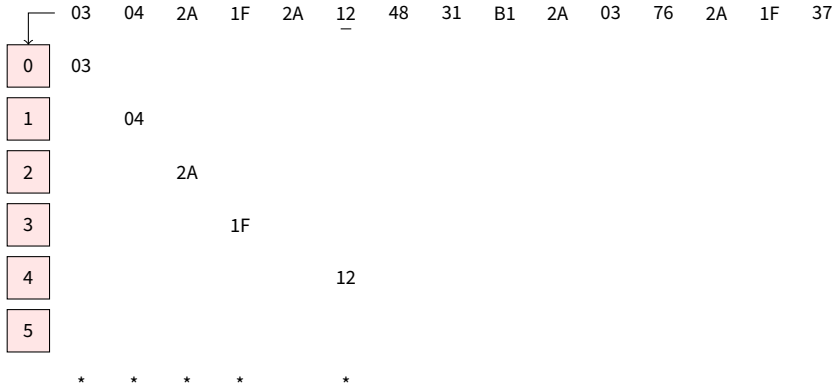
LEAST FREQUENTLY USED - LFU



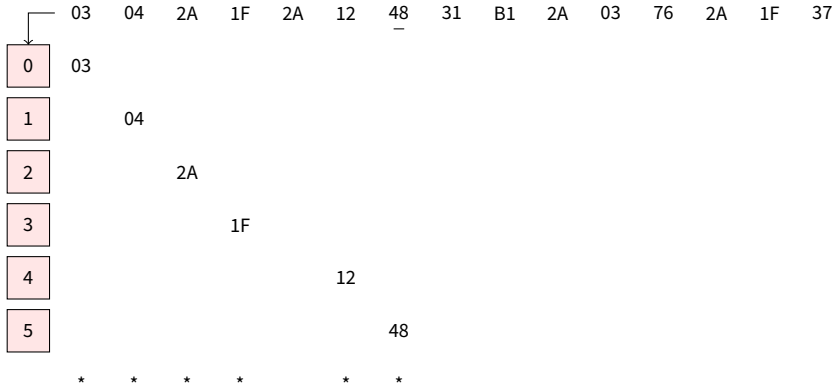
LEAST FREQUENTLY USED - LFU



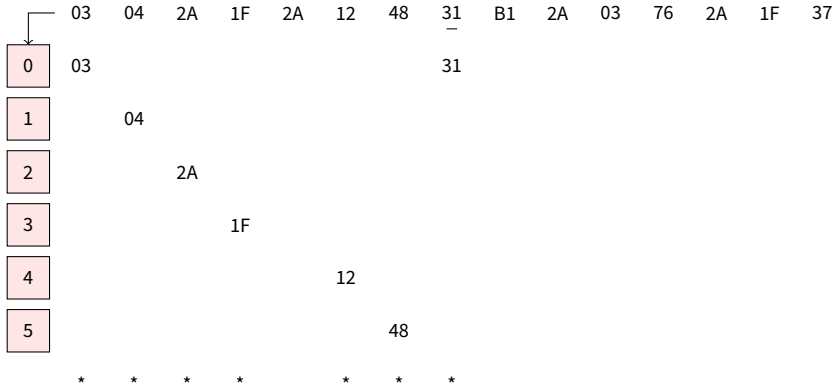
LEAST FREQUENTLY USED - LFU



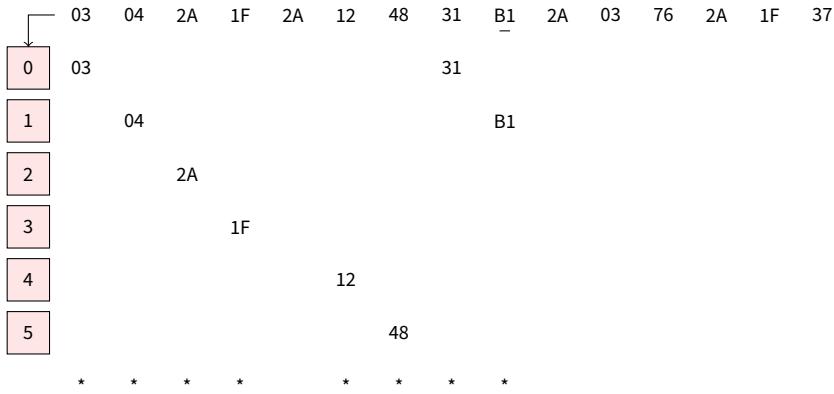
LEAST FREQUENTLY USED - LFU



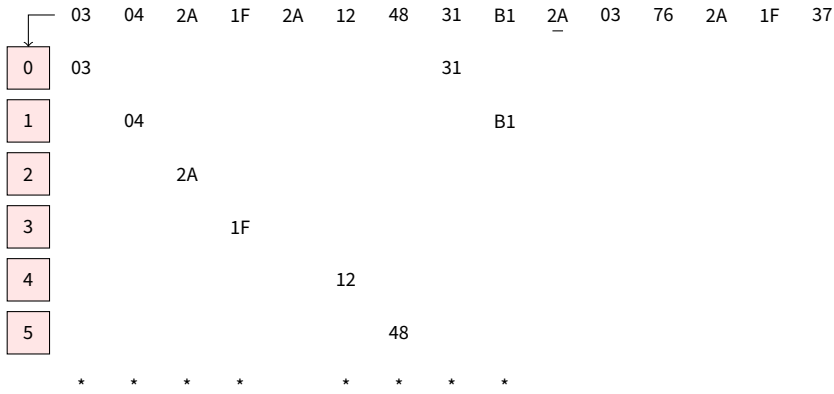
LEAST FREQUENTLY USED - LFU



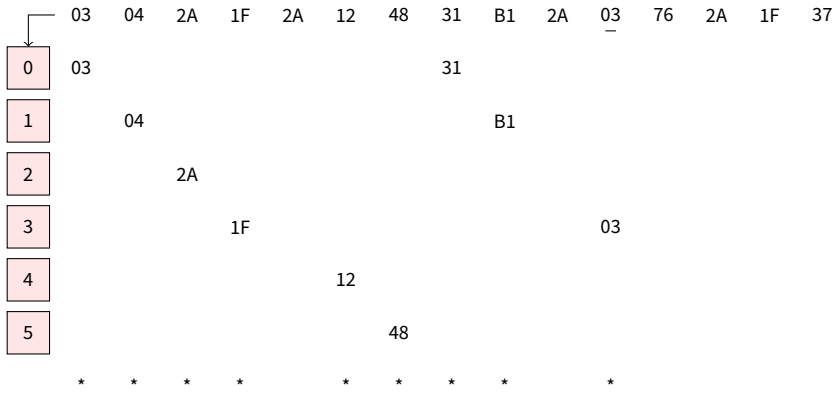
LEAST FREQUENTLY USED - LFU



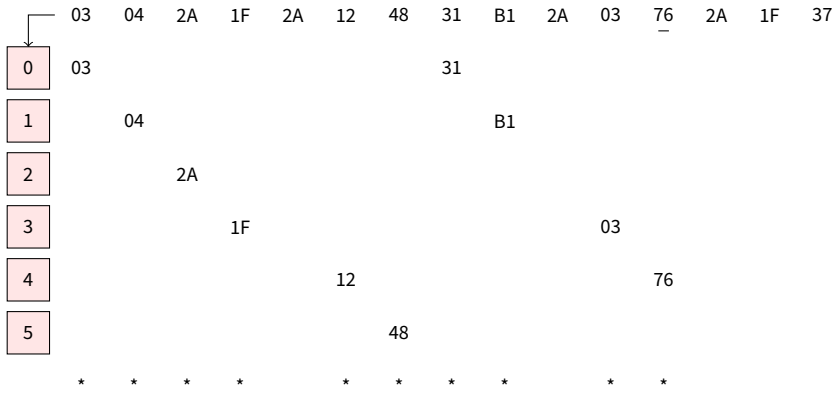
LEAST FREQUENTLY USED - LFU



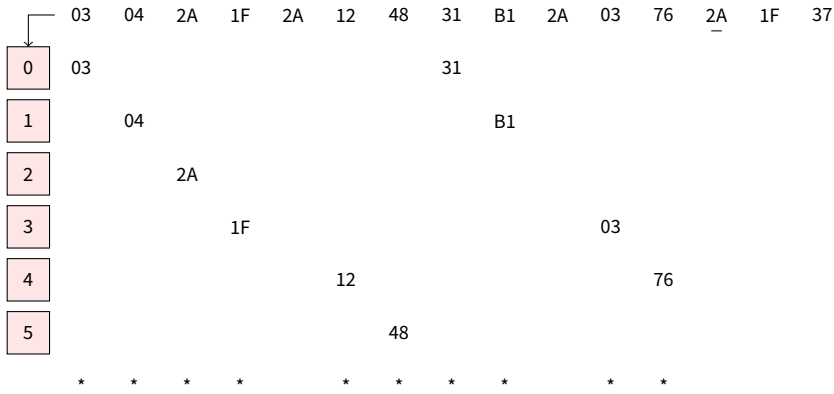
LEAST FREQUENTLY USED - LFU



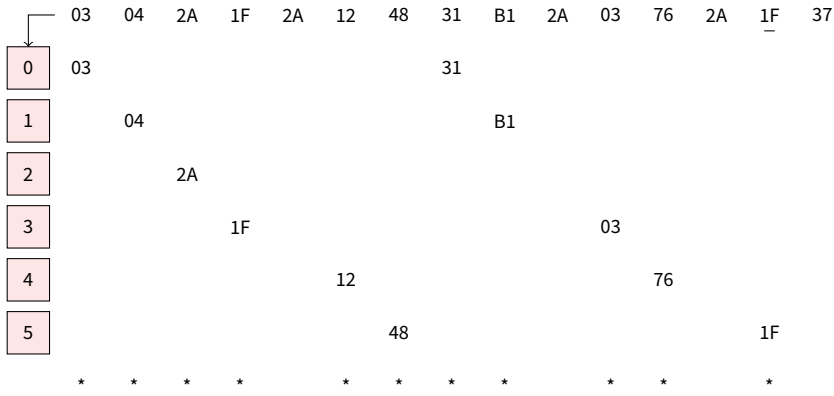
LEAST FREQUENTLY USED - LFU



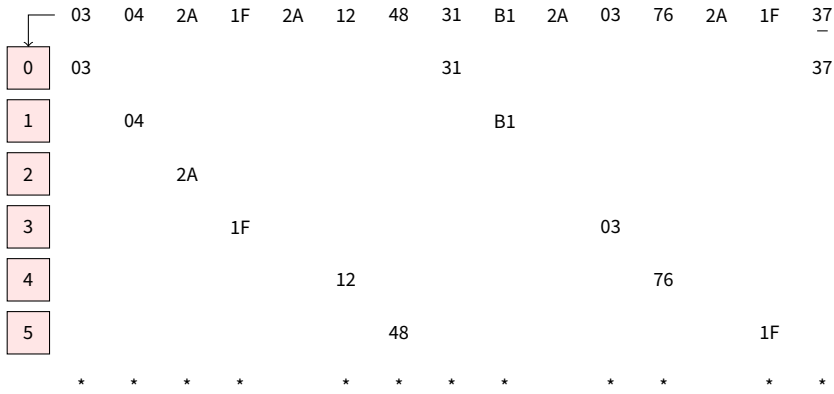
LEAST FREQUENTLY USED - LFU



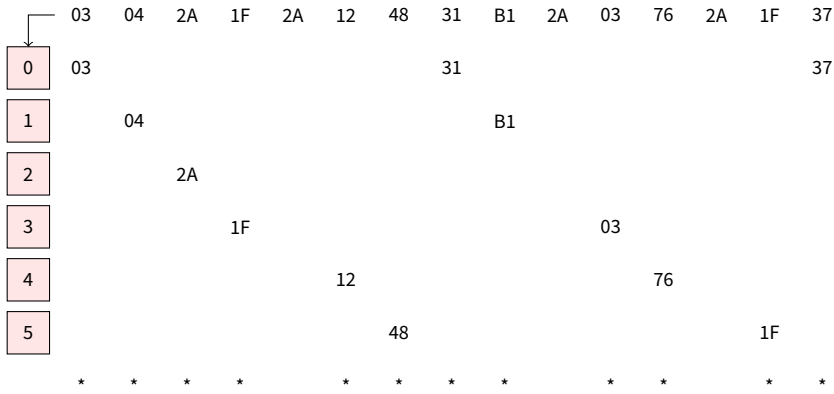
LEAST FREQUENTLY USED - LFU



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

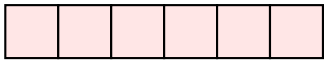
LEAST FREQUENTLY USED - LFU

- **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 référencée) 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$_r$



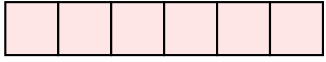
Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



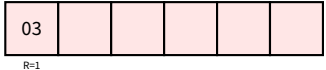
Page	R	M	Date
03	0	0	
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

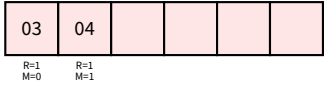


Page	R	M	Date
03	1	0	1
04	0	0	
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



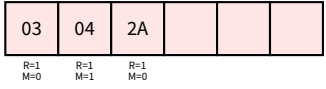
Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	0	0	
2A	0	0	
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

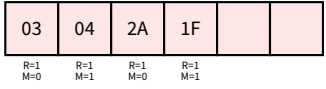


Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	0	0	
2A	1	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

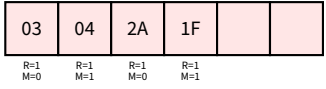


Page	R	M	Date
03	1	0	1
04	1	1	2
12	0	0	
1F	1	1	4
2A	1	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



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 référencée) 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

RESET

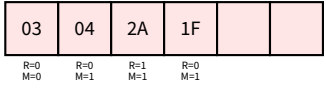
Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	
1F	0	1	4
2A	0	0	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

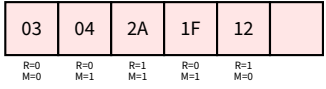


Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



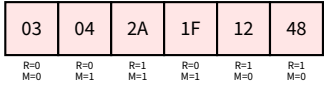
Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	0	0	
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



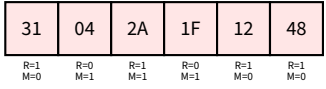
Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	0	0	
37	0	0	
48	1	0	7
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



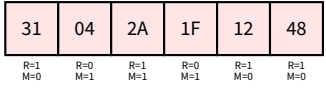
Page	R	M	Date
03	0	0	1
04	0	1	2
12	1	0	6
1F	0	1	4
2A	1	1	3
31	1	0	8
37	0	0	
48	1	0	7
76	0	0	
B1	0	0	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



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 référencée) 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

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	

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

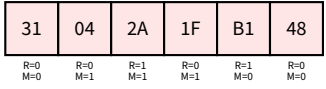
03r 04w 2Ar 1Fw 2Aw 12r 48r 31r B1r 2Ar 03w 76r 2Aw 1Fw 37r

Page	R	M	Date
03	0	0	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 référencée) 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



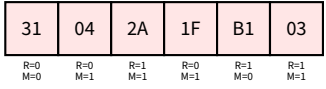
Page	R	M	Date
03	0	0	1
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	1	0	9

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) et M (page modifiée)
  - Lecture ou écriture → R=1 ; écriture → M=1
  - Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) >(R=0,M=0)
  - FIFO en cas d'égalité
  - Tous les R sont remis à 0 chaque K cycles

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



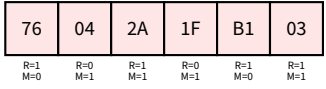
Page	R	M	Date
03	1	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	
B1	1	0	9

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) et M (page modifiée)
  - Lecture ou écriture → R=1 ; écriture → M=1
  - Priorité : (R=1,M=1) > (R=1,M=0) > (R=0,M=1) >(R=0,M=0)
  - FIFO en cas d'égalité
  - Tous les R sont remis à 0 chaque K cycles

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

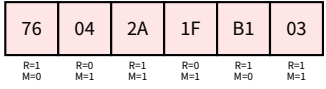


Page	R	M	Date
03	1	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	1	0	12
B1	1	0	9

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



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 référencée) 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

RESET

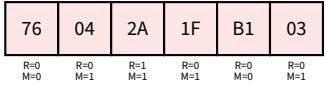
Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	0	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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



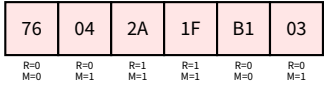
Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	0	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

CentraleSupélec
université PARIS-SACLAY

NOT RECENTLY USED - NRU

- **Principe**
  - 2 bits : R (page référencée) 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

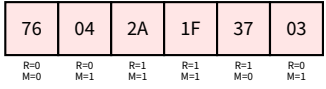


Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	1	1	4
2A	1	1	3
31	0	0	8
37	0	0	
48	0	0	7
76	0	0	12
B1	0	0	9

NOT RECENTLY USED - NRU

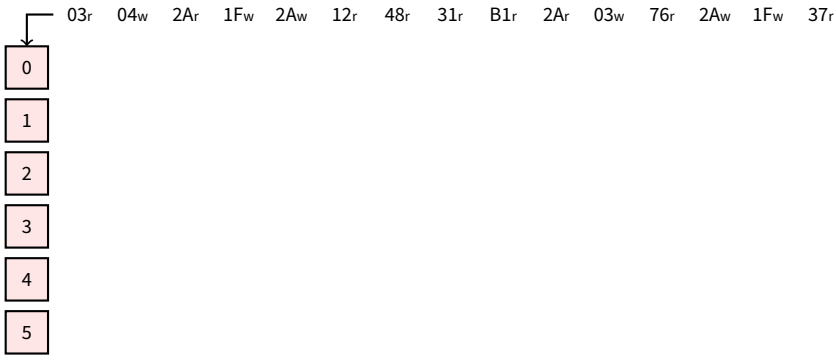
- **Principe**
  - 2 bits : R (page référencée) et M (page modifiée)
  - Lecture ou écriture $\rightarrow$ R=1 ; écriture $\rightarrow$ M=1
  - Priorité : (R=1,M=1) $>$ (R=1,M=0) $>$ (R=0,M=1) $>$(R=0,M=0)
  - FIFO en cas d'égalité
  - Tous les R sont remis à 0 chaque K cycles

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

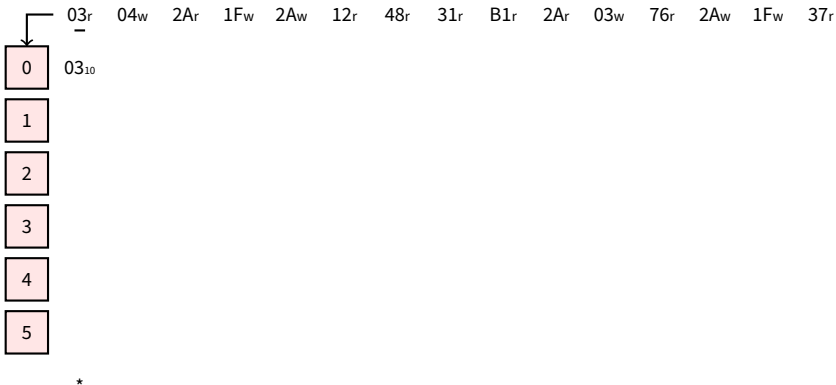


Page	R	M	Date
03	0	1	11
04	0	1	2
12	0	0	6
1F	1	1	4
2A	1	1	3
31	0	0	8
37	1	0	15
48	0	0	7
76	0	0	12
B1	0	0	9

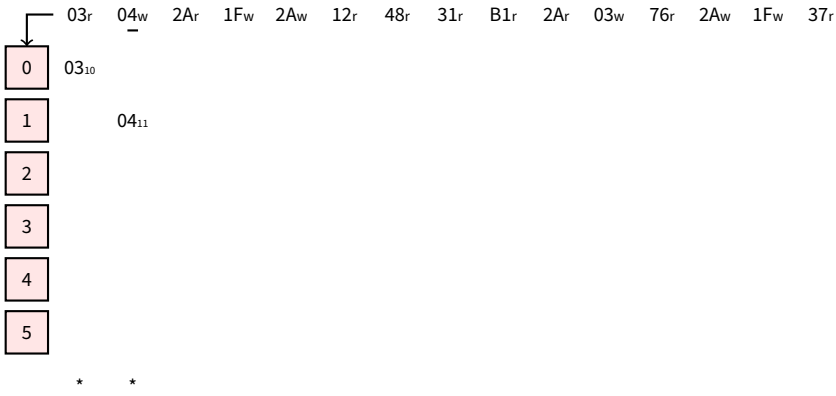
NOT RECENTLY USED - NRU



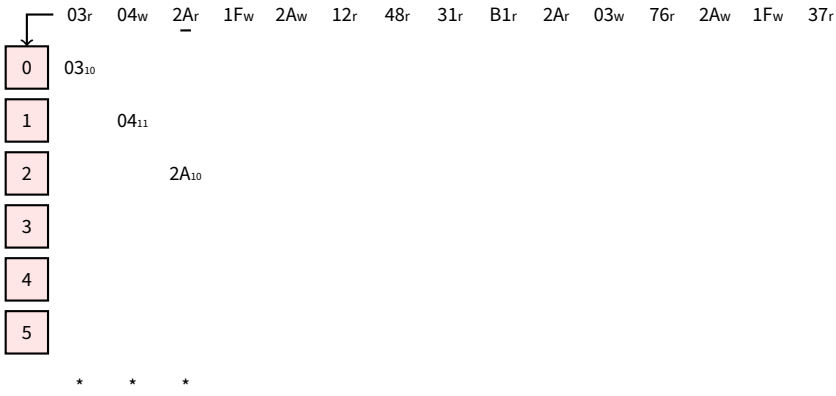
NOT RECENTLY USED - NRU



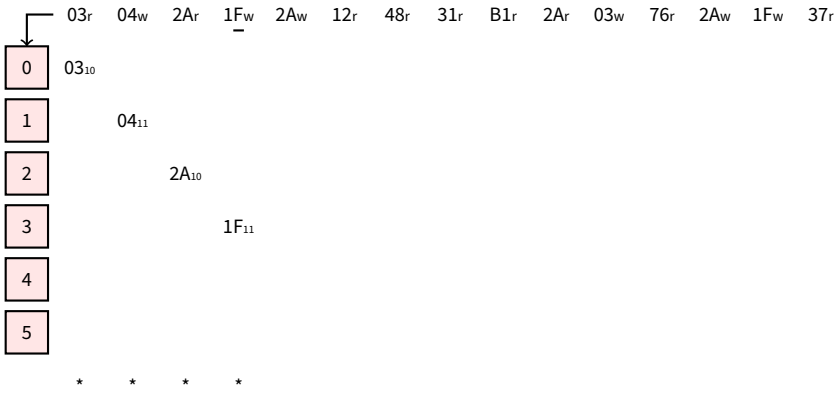
NOT RECENTLY USED - NRU



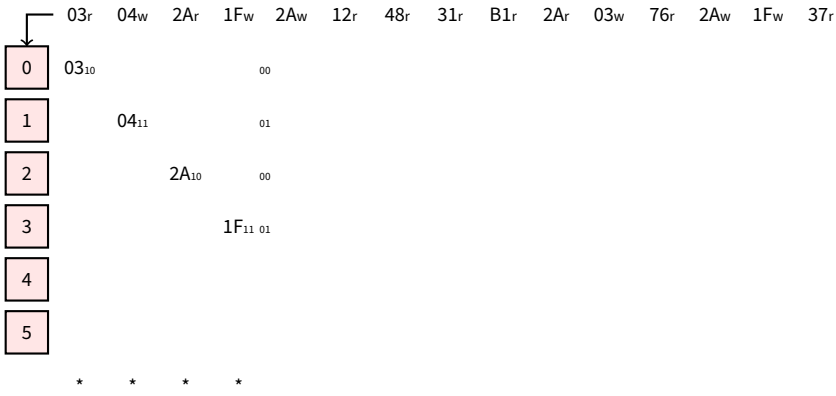
NOT RECENTLY USED - NRU



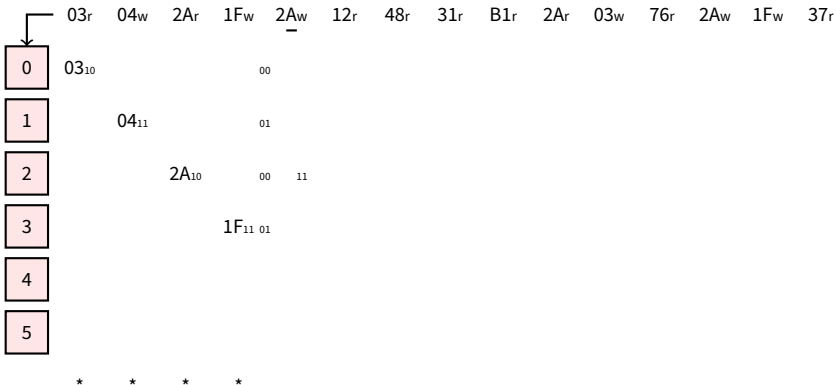
NOT RECENTLY USED - NRU



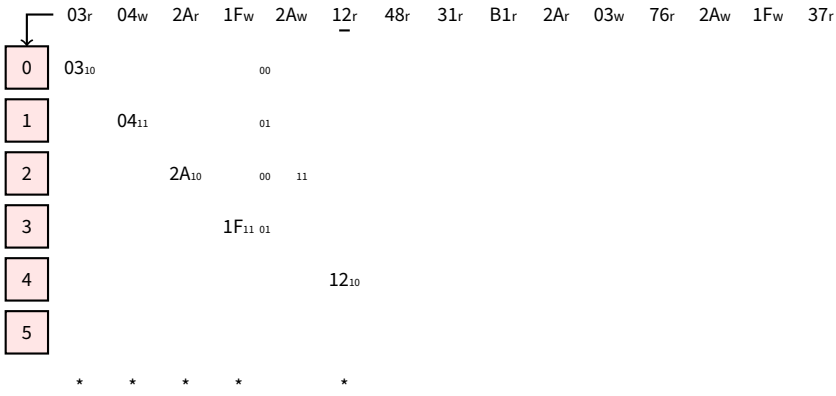
NOT RECENTLY USED - NRU



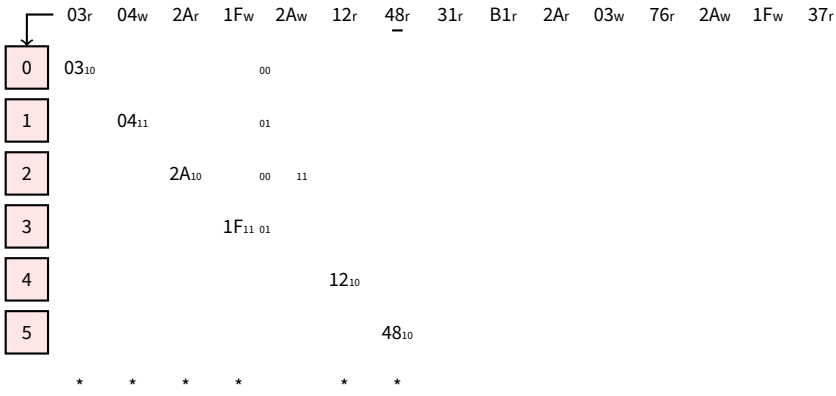
NOT RECENTLY USED - NRU



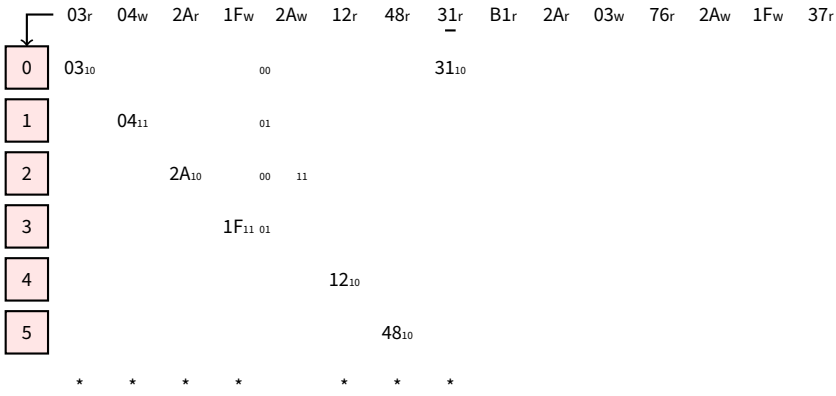
NOT RECENTLY USED - NRU



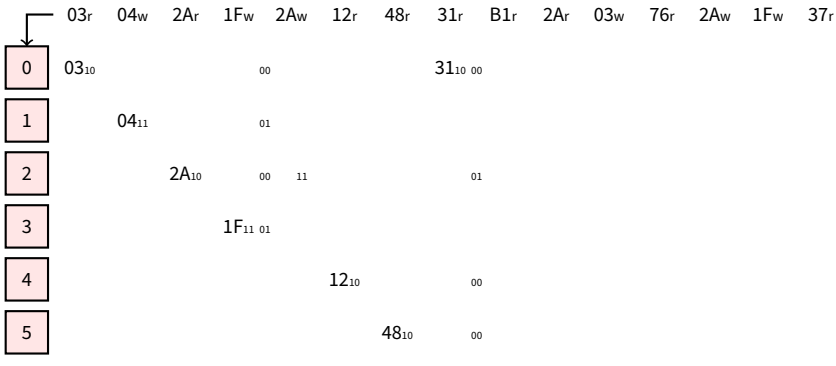
NOT RECENTLY USED - NRU



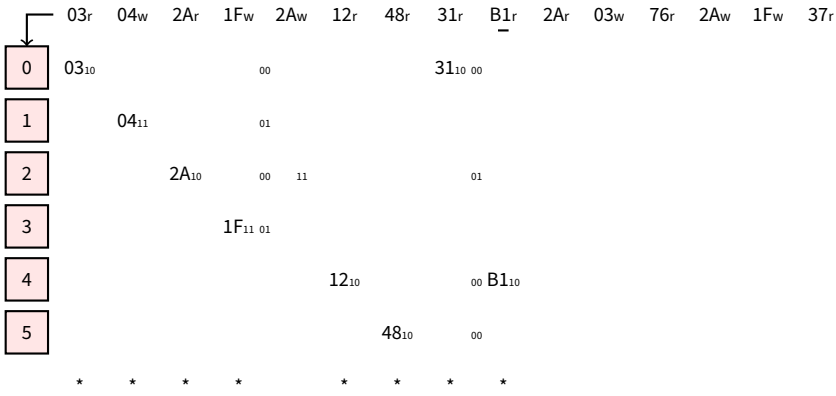
NOT RECENTLY USED - NRU



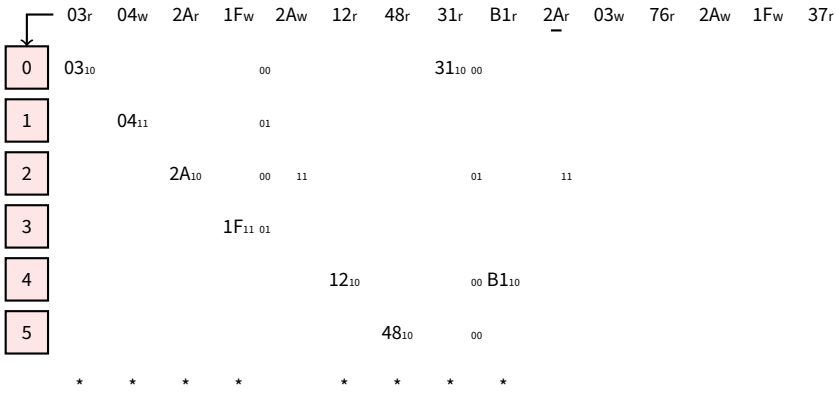
NOT RECENTLY USED - NRU



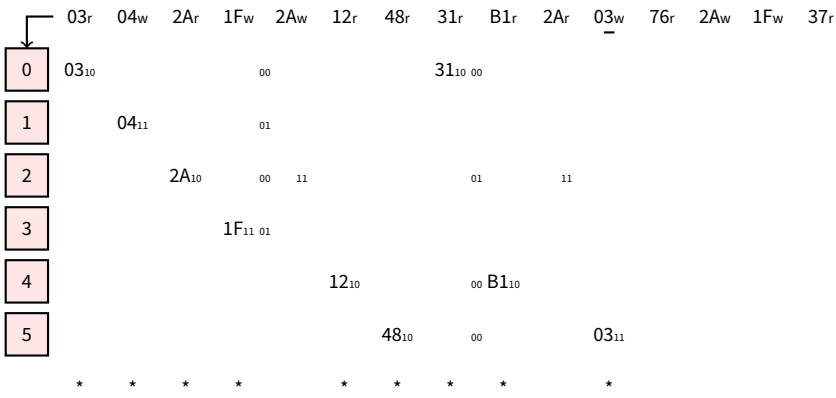
NOT RECENTLY USED - NRU



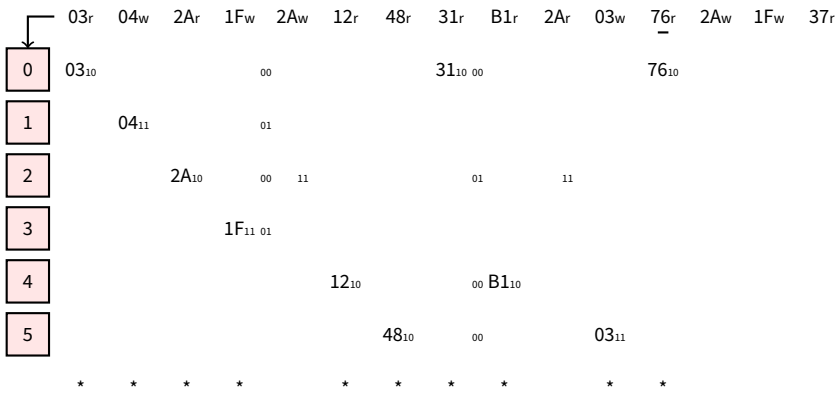
NOT RECENTLY USED - NRU



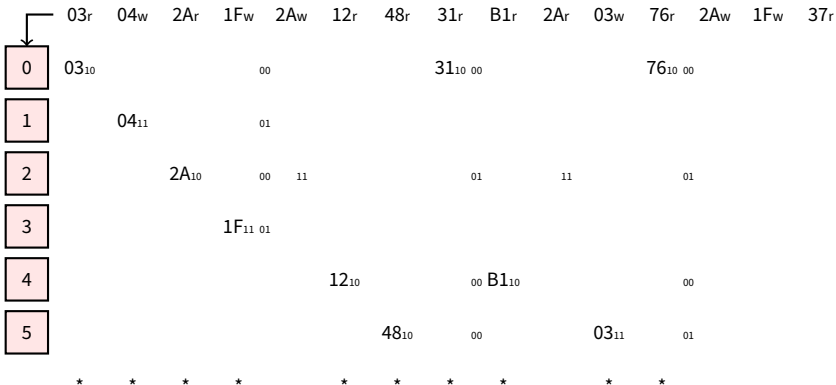
NOT RECENTLY USED - NRU



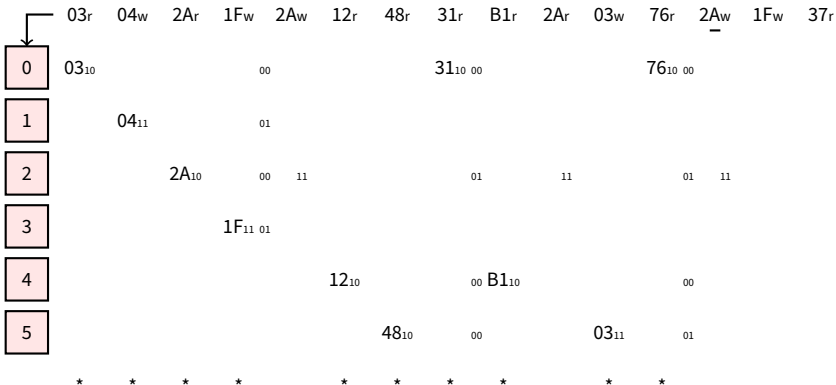
NOT RECENTLY USED - NRU



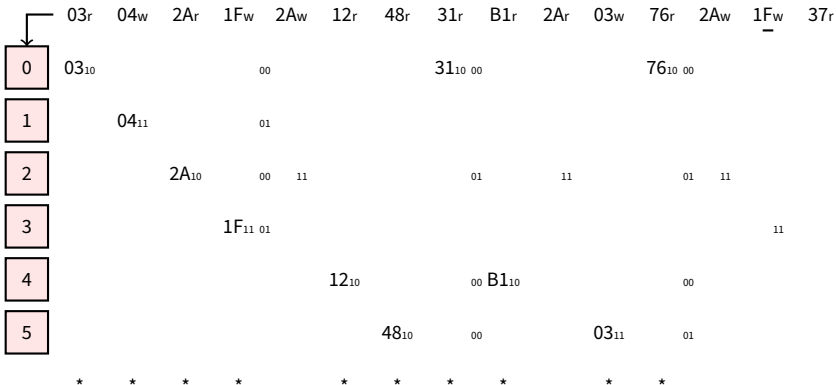
NOT RECENTLY USED - NRU



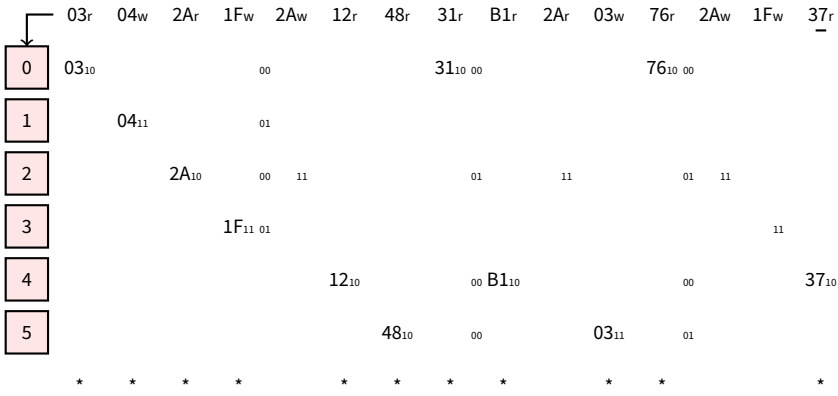
NOT RECENTLY USED - NRU



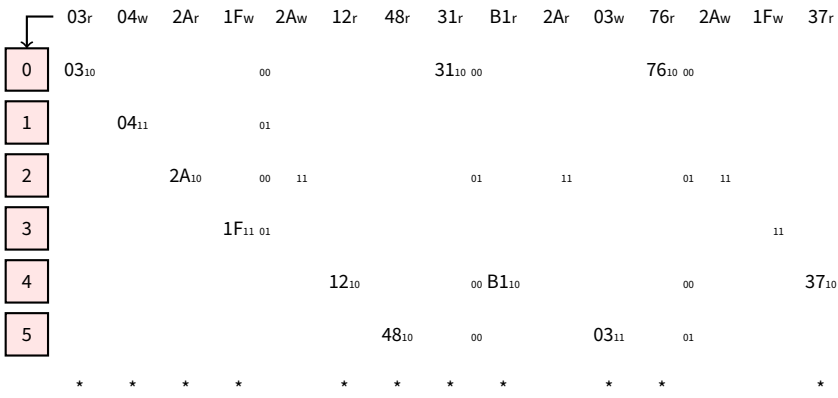
NOT RECENTLY USED - NRU



NOT RECENTLY USED - NRU



NOT RECENTLY USED - NRU



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

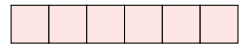
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

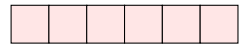
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



LEAST RECENTLY USED - LRU

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

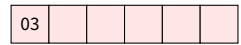
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



LEAST RECENTLY USED - LRU

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

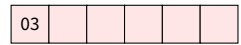
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04				

LEAST RECENTLY USED - LRU

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

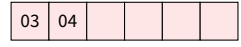
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A			

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A			

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ 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		

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A	1F		

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A	1F		

LEAST RECENTLY USED - LRU

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

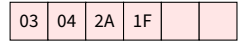
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A	1F	12	

LEAST RECENTLY USED - LRU

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

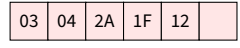
  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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



CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

03	04	2A	1F	12	48

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	04	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	04	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ 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

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	1F	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ 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

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	12	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	76	48

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	76	48

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	76	48

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	76	48

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ 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

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

31	B1	2A	03	76	1F

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\longrightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

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

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

37	B1	2A	03	76	1F

CentraleSupélec
université PARIS-SACLAY

LEAST RECENTLY USED - LRU

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

  $\rightarrow$ remettre la ligne i à 1 puis la colonne i à 0

➠ Cadre le plus ancien (à utiliser) =

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

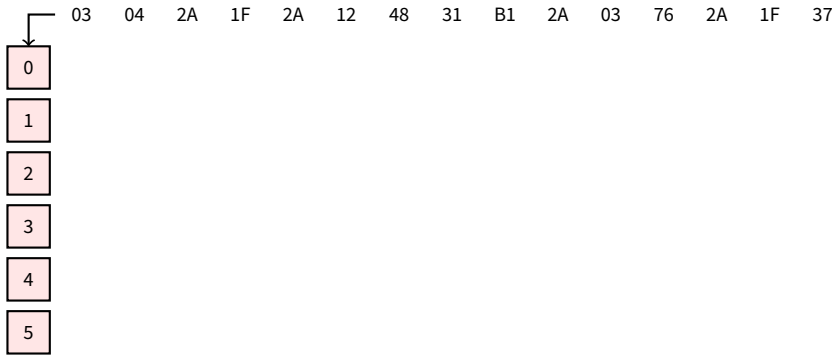
	0	1	2	3	4	5
0		1	1	1	1	1
1			0	0	0	0
2				1	1	0
3					0	0
4						0
5						

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

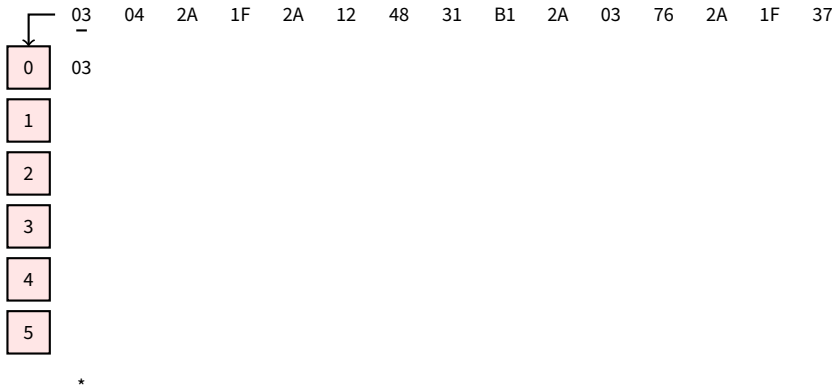
37	B1	2A	03	76	1F

CentraleSupélec
université PARIS-SACLAY

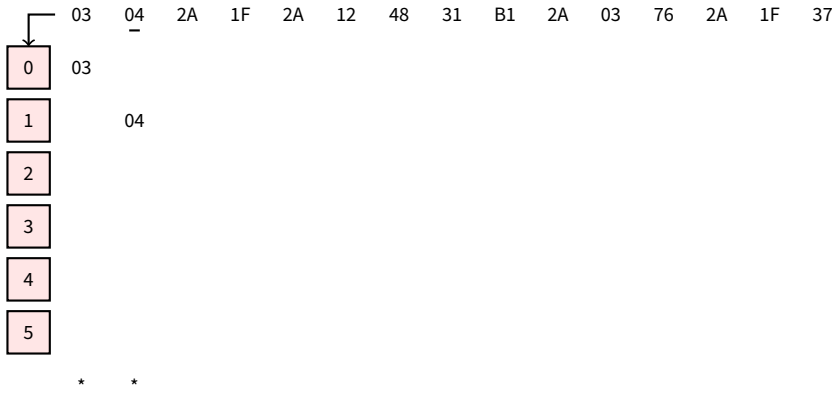
LEAST RECENTLY USED - LRU



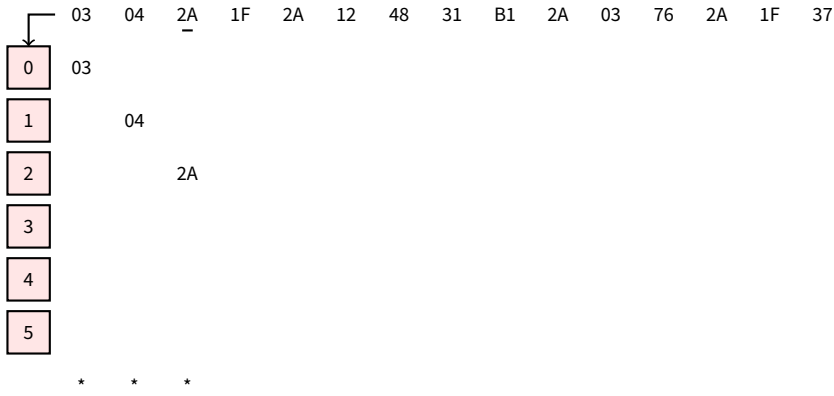
LEAST RECENTLY USED - LRU



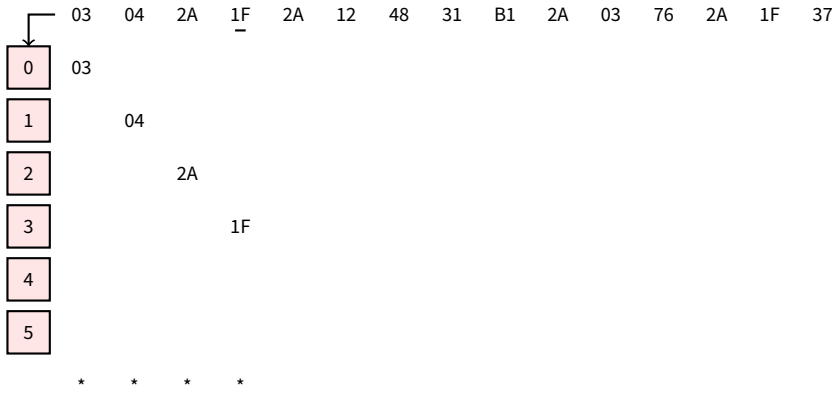
LEAST RECENTLY USED - LRU



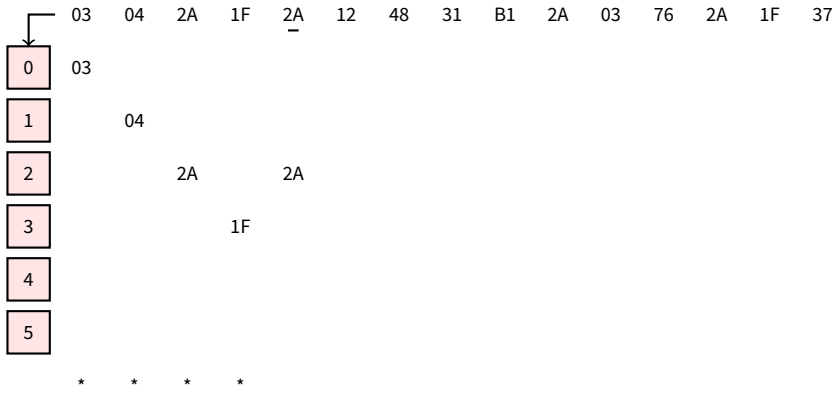
LEAST RECENTLY USED - LRU



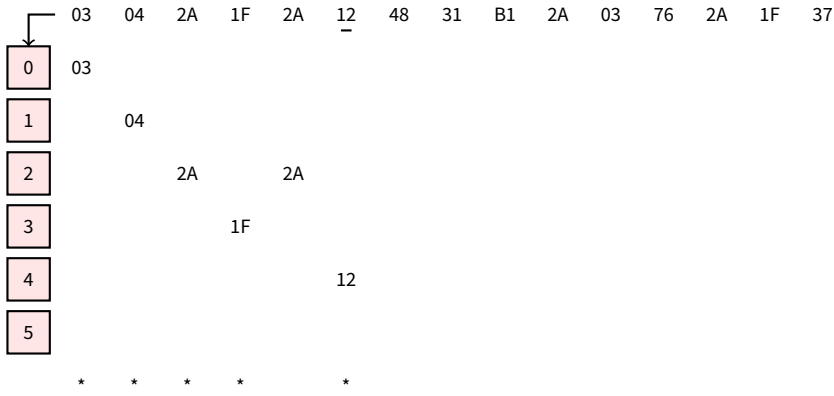
LEAST RECENTLY USED - LRU



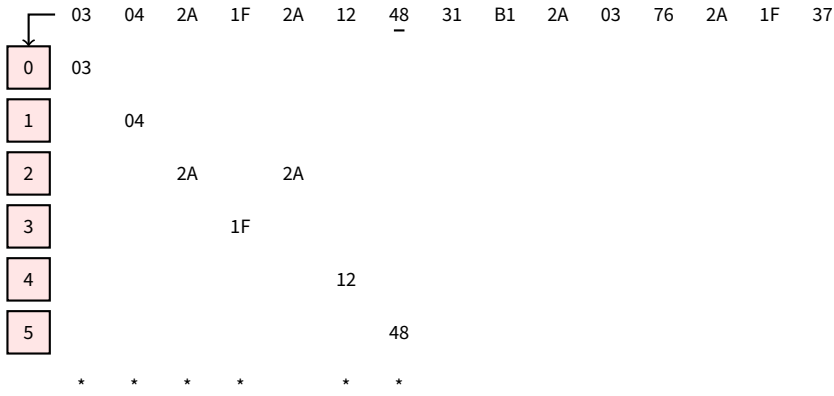
LEAST RECENTLY USED - LRU



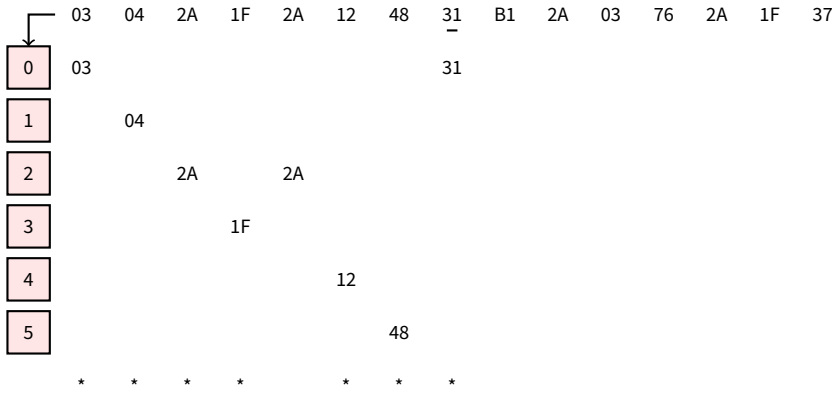
LEAST RECENTLY USED - LRU



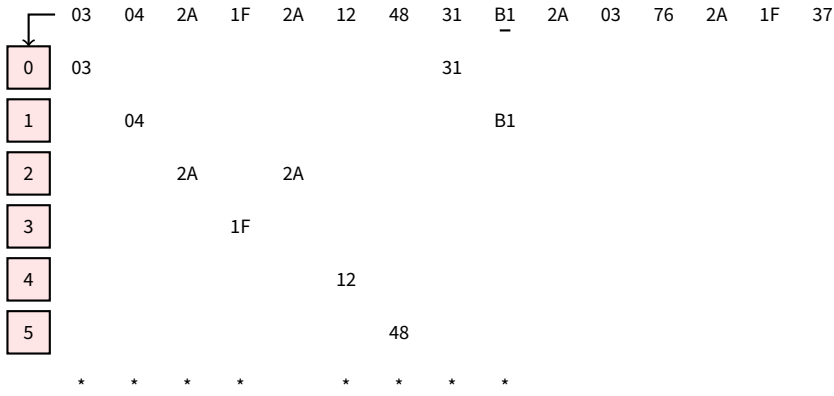
LEAST RECENTLY USED - LRU



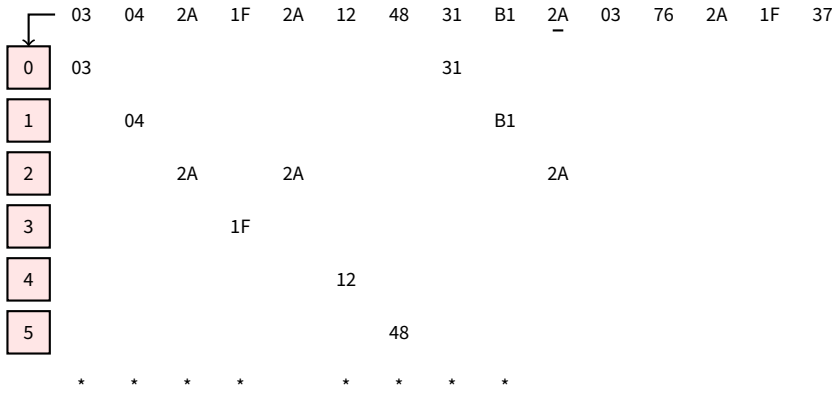
LEAST RECENTLY USED - LRU



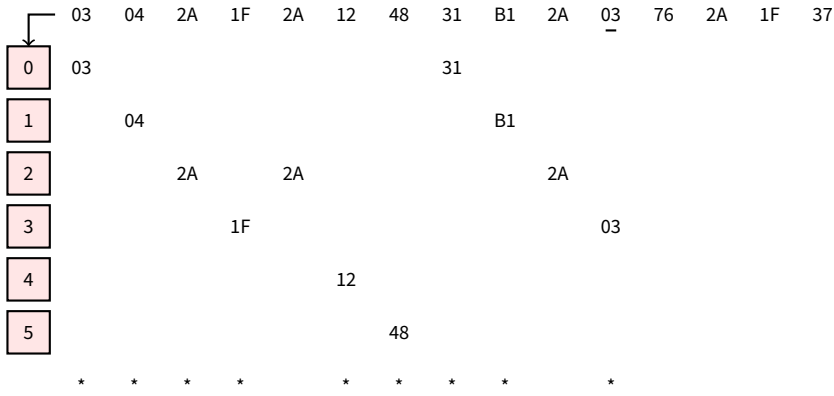
LEAST RECENTLY USED - LRU



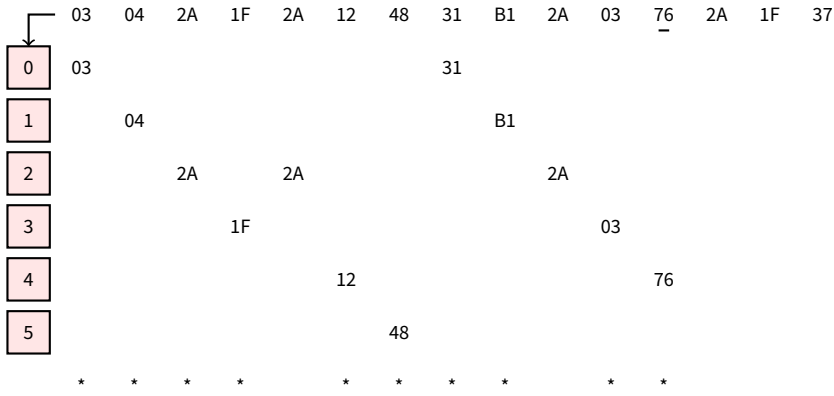
LEAST RECENTLY USED - LRU



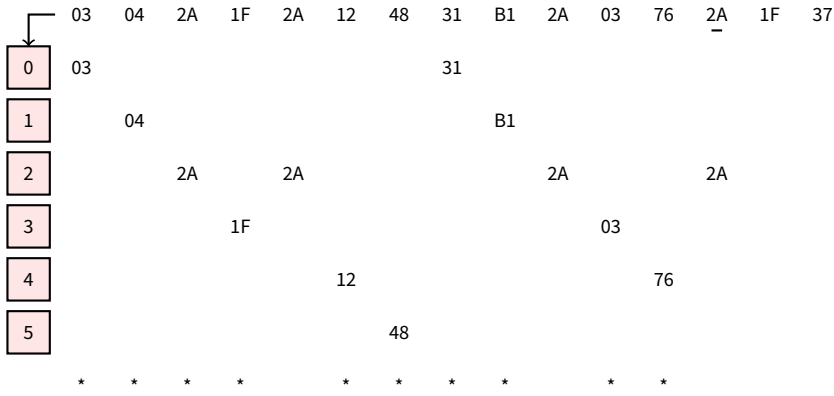
LEAST RECENTLY USED - LRU



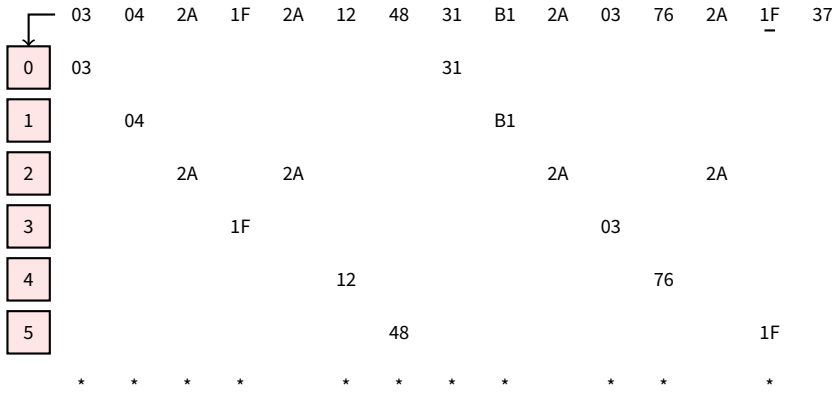
LEAST RECENTLY USED - LRU



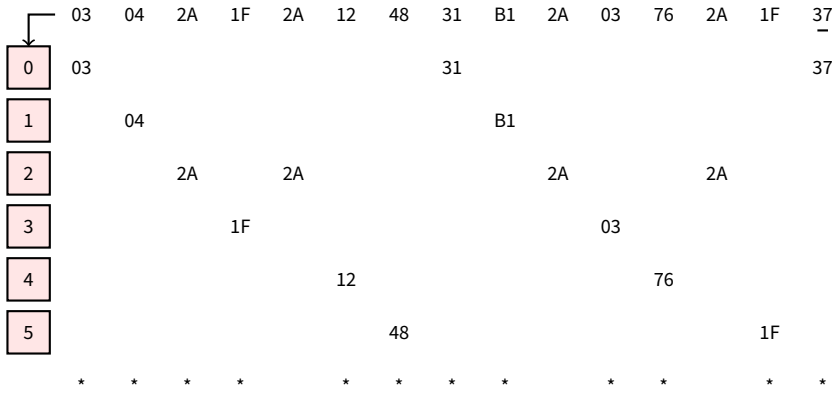
LEAST RECENTLY USED - LRU



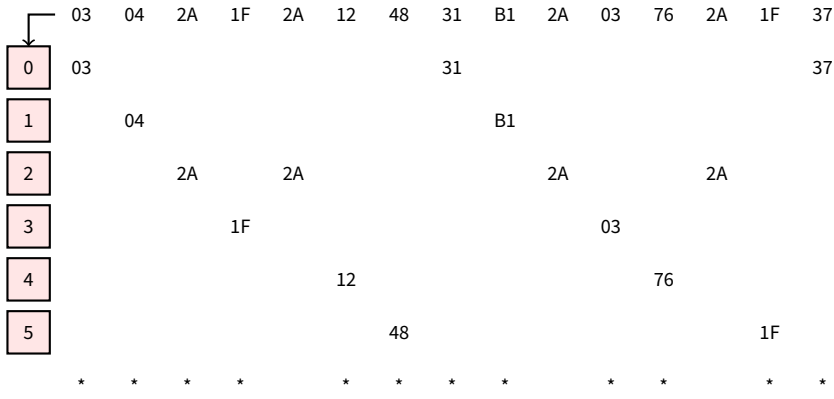
LEAST RECENTLY USED - LRU



LEAST RECENTLY USED - LRU



LEAST RECENTLY USED - LRU



Total : 12 défauts

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 $\longrightarrow$ **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 $\longrightarrow$ **mémoire segmentée**
  - Découper en tenant compte de la structure du processus (code + données)

OUTLINE

Back to the begin - Back to the outline

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 :

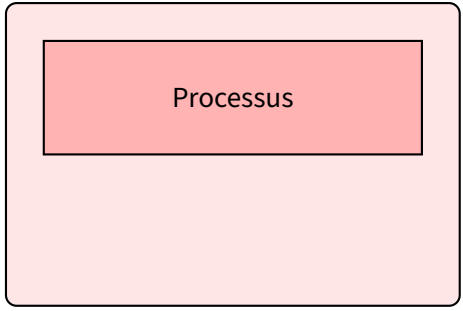
- Pile
- Table des symboles
- Programme principal
- Fonction
- Bibliothèque

MÉMOIRE SEGMENTÉE

Structurer la mémoire en blocs de données/routines indépendants :

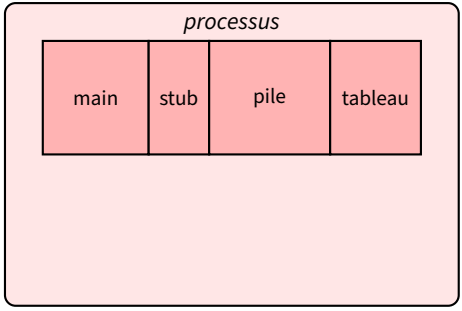
- Pile
- Table des symboles
- Programme principal
- Fonction
- Bibliothèque
- …

STRUCTURE EN SEGMENTS



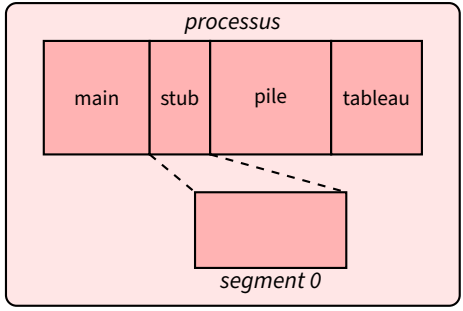
Espace d’adressage logique

STRUCTURE EN SEGMENTS



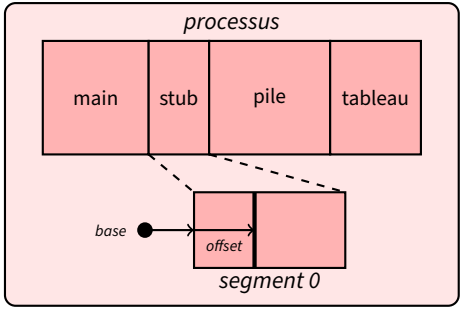
Espace d’adressage logique

STRUCTURE EN SEGMENTS



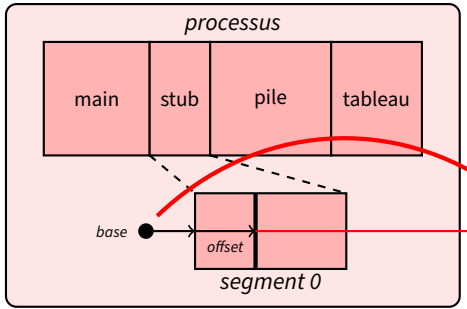
Espace d'adressage logique

STRUCTURE EN SEGMENTS



Espace d'adressage logique

STRUCTURE EN SEGMENTS



Espace d'adressage logique

Routine printf x86 (stdio)

```
mov eax, 4 ; Appel de sys_write
mov ebx, 1 ; Sortie standard STDOUT
mov ecx, Buffer ; Chaine à afficher
mov edx, BufferSize ; Taille de la
chaine
int 80h ; Interruption du kernel
mov eax, 1 ; Appel de sys_exit
mov ebx, 0 ; Code de retour
int 80h ; Interruption du kernel
```

CONSTRUCTION DES SEGMENTS

CONSTRUCTION DES SEGMENTS

- **À la compilation**
  - dépendant du langage
  - chaque segment est référencé par un numéro

    $\longrightarrow$ 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** $\longrightarrow$ Memory Management Unit
  - Transformer segment + décalage en adresse physique

ADRESSAGE

- **Adresse logique**
  - Numéro de segment (*selecteur*)+ décalage
- **Résolution** $\longrightarrow$ Memory Management Unit
  - Transformer segment + décalage en adresse physique
  - Vérifier que le décalage ne sort pas du segment

$\longrightarrow$ *erreur de segmentation*

CentraleSupélec
université PARIS-SACLAY

RÉSOLUTION D'ADRESSE

RÉSOLUTION D'ADRESSE

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

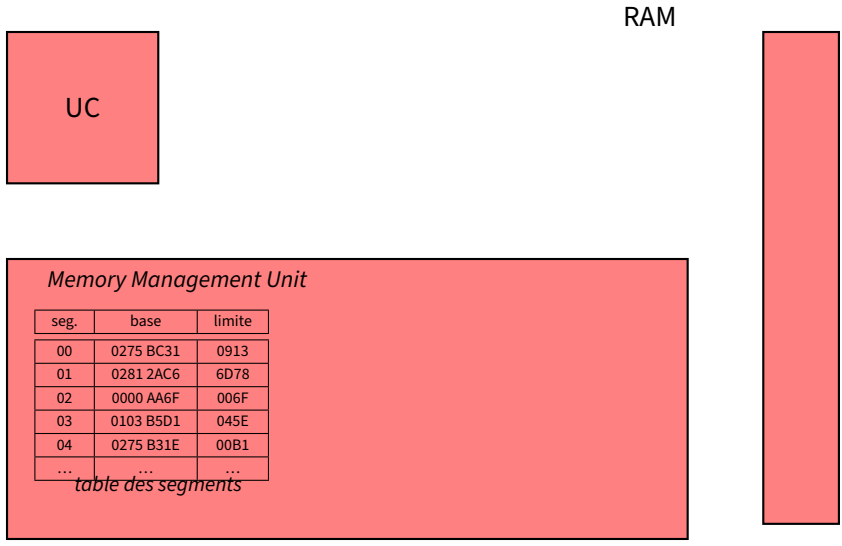
RÉSOLUTION D'ADRESSE

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

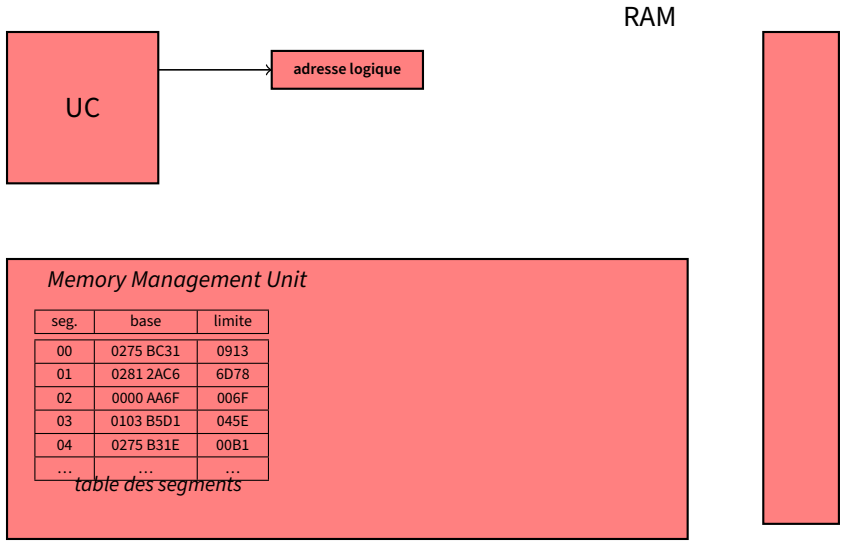
- **Table des segments** : pour chaque segment :
  - Base = adresse physique de départ
  - Limite = taille du segment

. . .aussi appelée : table des descripteurs

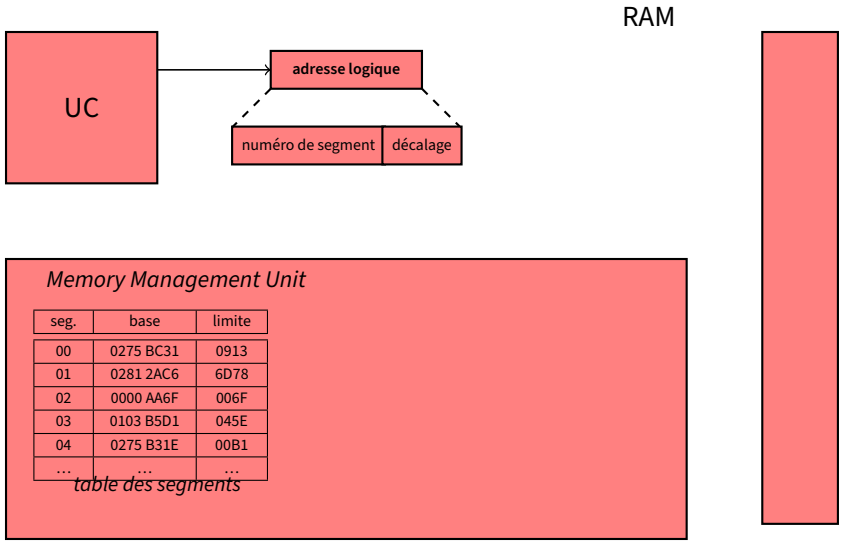
RÉSOLUTION D'ADRESSE



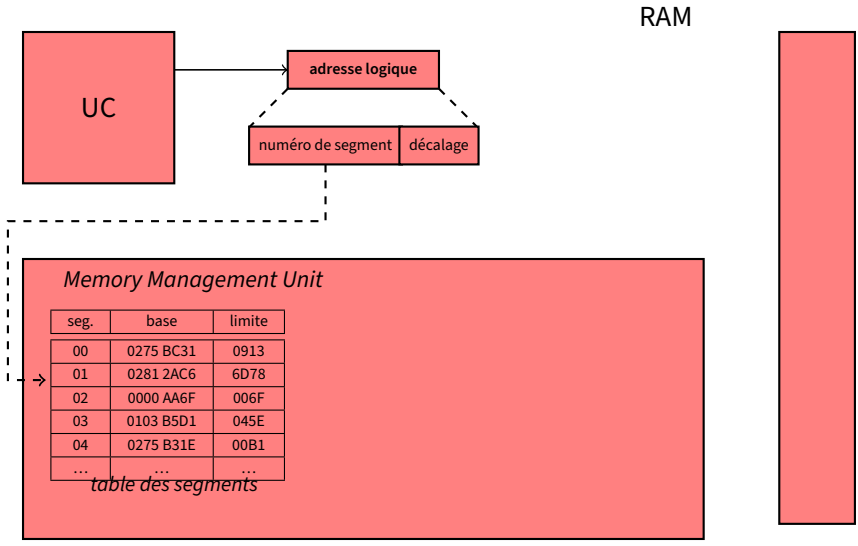
RÉSOLUTION D'ADRESSE



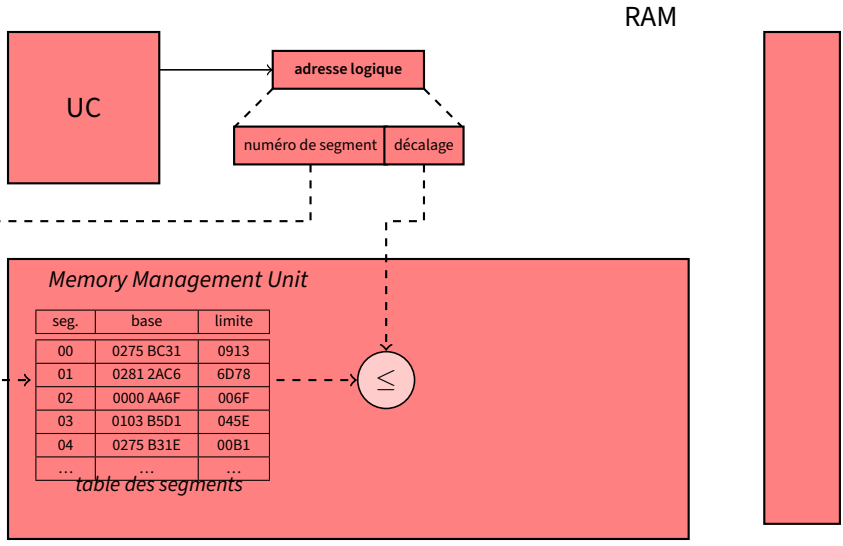
RÉSOLUTION D'ADRESSE



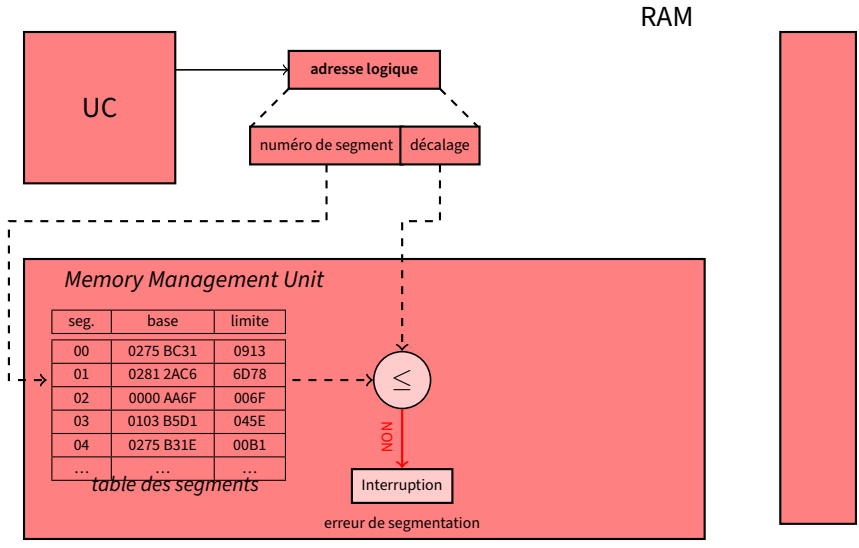
RÉSOLUTION D'ADRESSE



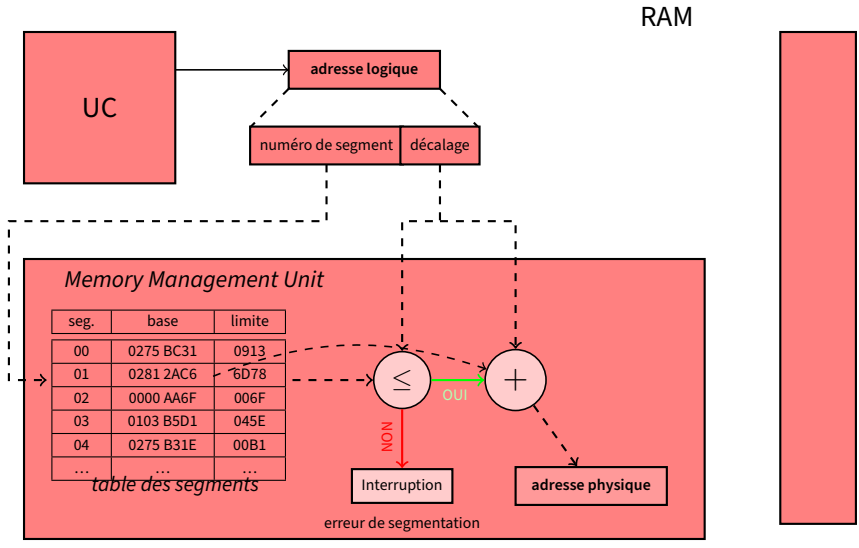
RÉSOLUTION D'ADRESSE



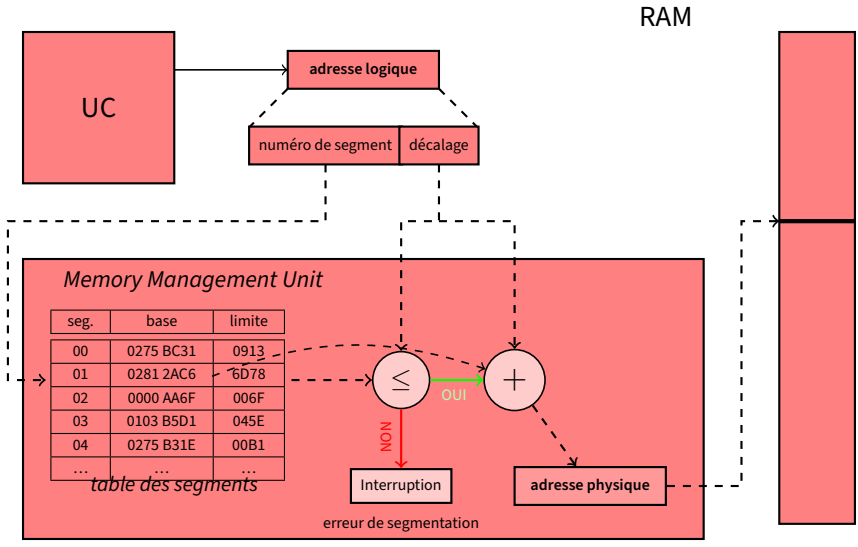
RÉSOLUTION D'ADRESSE



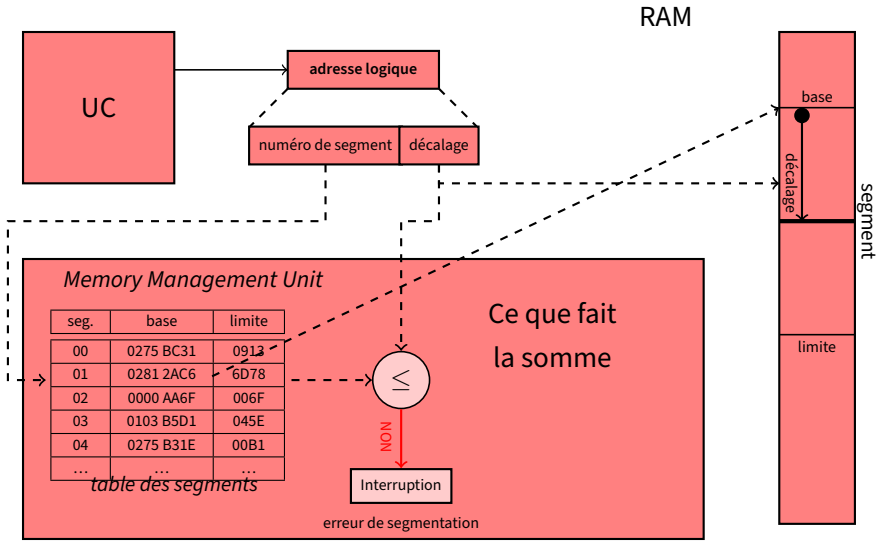
RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



PARTAGE DE SEGMENTS

PARTAGE DE SEGMENTS

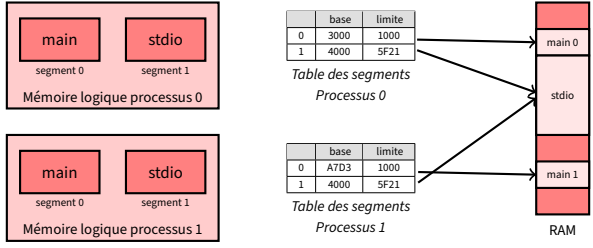
- Un même segment (de code) utilisé par plusieurs processus

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

  $\longrightarrow$ 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

  $\longrightarrow$ Adresse définie lors du premier chargement

✔ **Protection**

➠ Par segment **et** par processus !

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

  $\longrightarrow$ 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

  $\longrightarrow$ Adresse définie lors du premier chargement

✔ **Protection**

➠ Par segment **et** par processus !

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

  $\longrightarrow$ 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

CentraleSupélec
université PARIS-SACLAY

QUELQUES LIMITES

QUELQUES LIMITES

✘ **Taille des segments variables**

➠ On retombe sur le problème de l'allocation contigüe :

- Fragmentation $\longrightarrow$ perte de mémoire
- Défragmentation $\longrightarrow$ perte de temps à l'exécution

QUELQUES LIMITES

✘ **Taille des segments variables**

➠ On retombe sur le problème de l'allocation contigüe :

- Fragmentation $\longrightarrow$ perte de mémoire
- Défragmentation $\longrightarrow$ perte de temps à l'exécution

✔ **Solution** $\longrightarrow$ **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 + décalage$_1$

  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 + décalage$_1$

  résolu par segmentation donne :
- Adresse linéaire = (répertoire +)^1 page + décalage$_2$

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 + décalage$_1$

  résolu par segmentation donne :
- Adresse linéaire = (répertoire +)^1 page + décalage$_2$
- Adresse physique = cadre de page + décalage$_2$

CentraleSupélec
université PARIS-SACLAY

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 + décalage$_1$

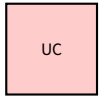
  résolu par segmentation donne :
- Adresse linéaire = (répertoire +)^1 page + décalage$_2$
- Adresse physique = cadre de page + décalage$_2$

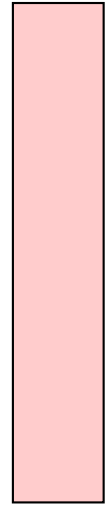
- **Mémory Management Unit**
  - Table des descripteurs : adresse logique $\longrightarrow$ adresse linéaire
  - Répertoire^1 + table des pages : adresse linéaire $\longrightarrow$ physique

^{1} si pagination à 2 niveaux

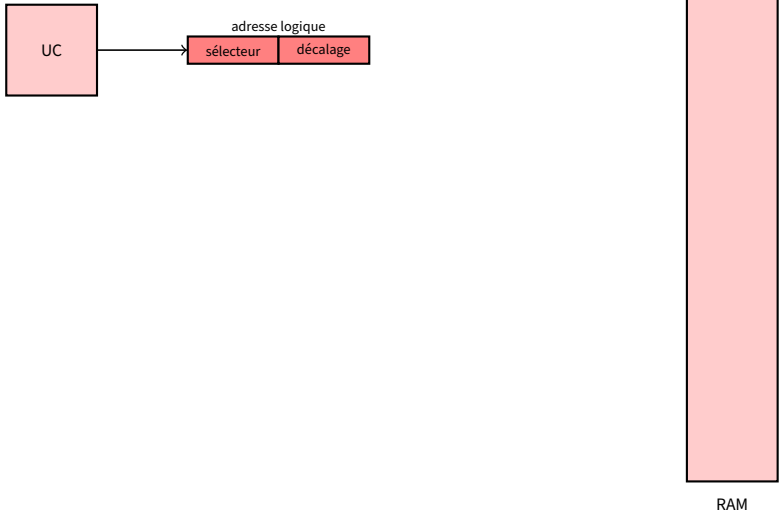
CentraleSupélec
université PARIS-SACLAY

IMPLÉMENTATION

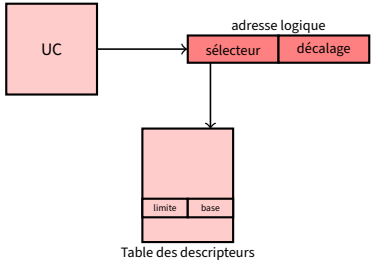


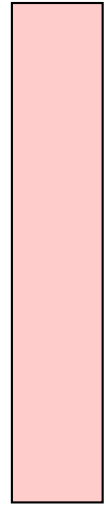


IMPLÉMENTATION

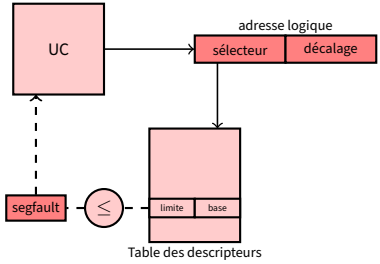


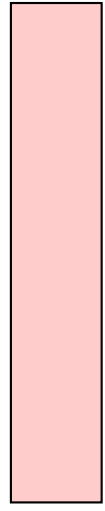
IMPLÉMENTATION



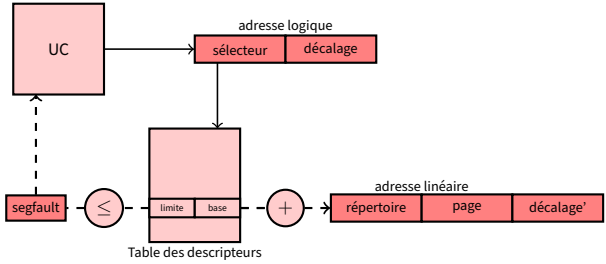


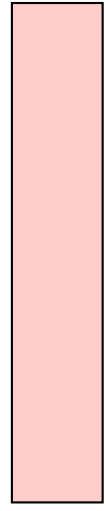
IMPLÉMENTATION



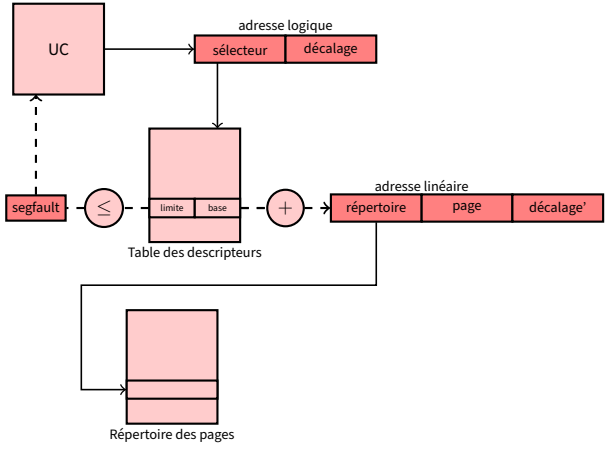


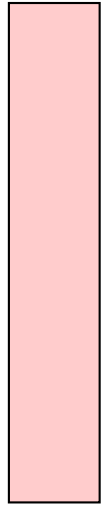
IMPLÉMENTATION





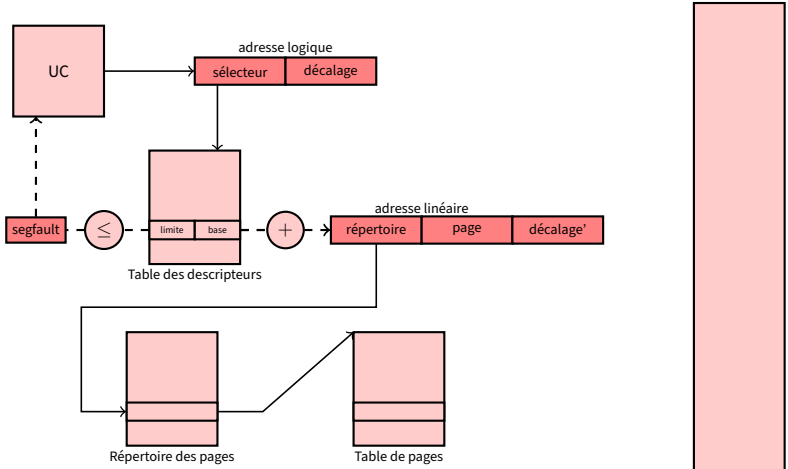
IMPLÉMENTATION





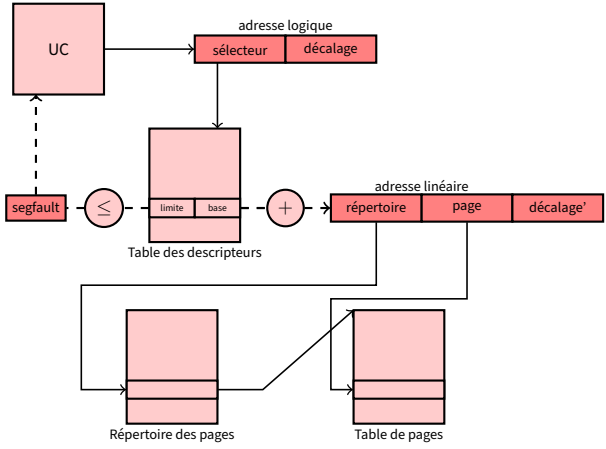
CentraleSupélec
université PARIS-SACLAY

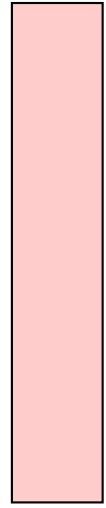
IMPLÉMENTATION



CentraleSupélec
université PARIS-SACLAY

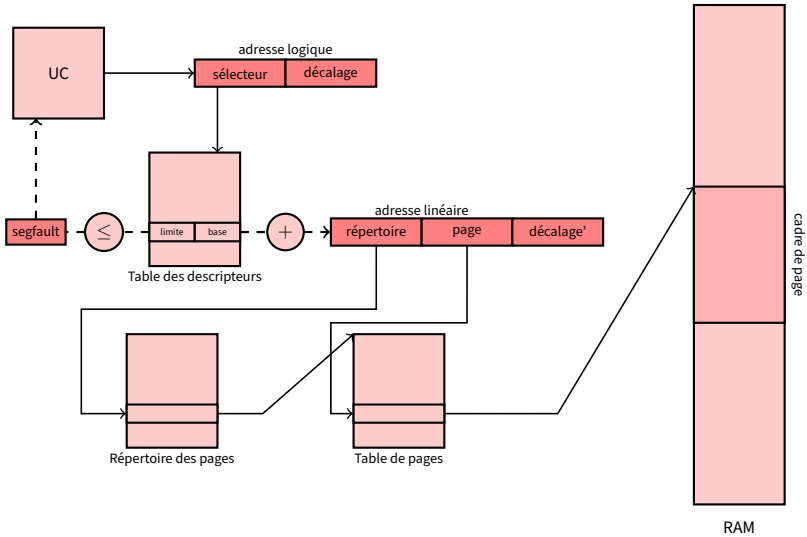
IMPLÉMENTATION



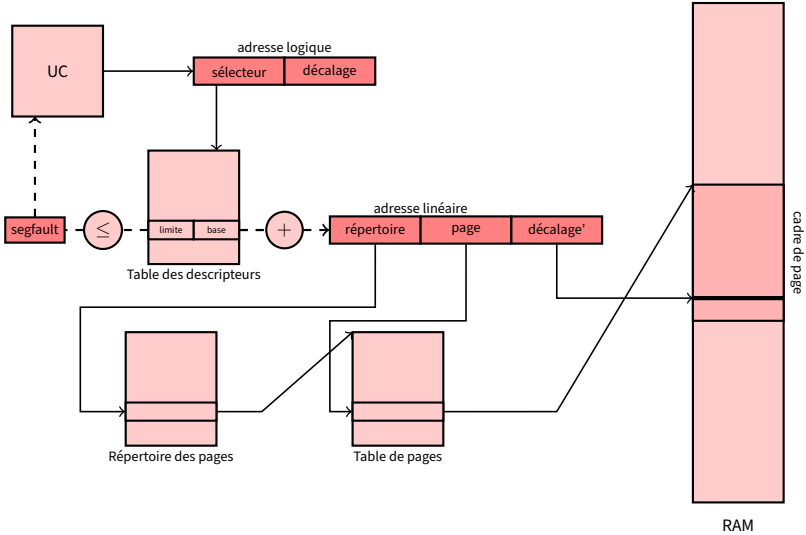


CentraleSupélec
université PARIS-SACLAY

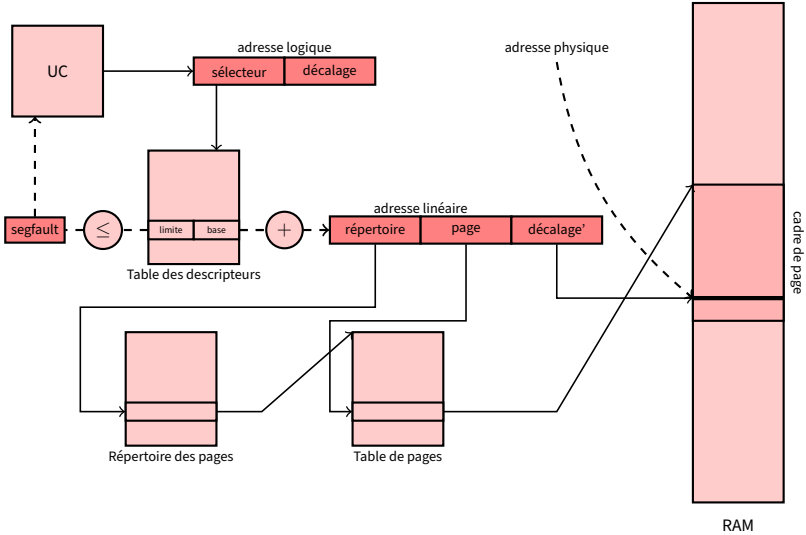
IMPLÉMENTATION



IMPLÉMENTATION



IMPLÉMENTATION



OUTLINE

Back to the begin - Back to the outline

LA SYNTHÈSE

LA SYNTHÈSE

- Gestion de la mémoire

LA SYNTHÈSE

- Gestion de la mémoire
- Allocation par partitions

LA SYNTHÈSE

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

LA SYNTHÈSE

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

LA SYNTHÈSE

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

CentraleSupélec
université PARIS-SACLAY

LA SYNTHÈSE

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

CentraleSupélec
université PARIS-SACLAY

THANK YOU

Back to the begin - Back to the outline