

# Systemes d'exploitation

## ENSISA 1A

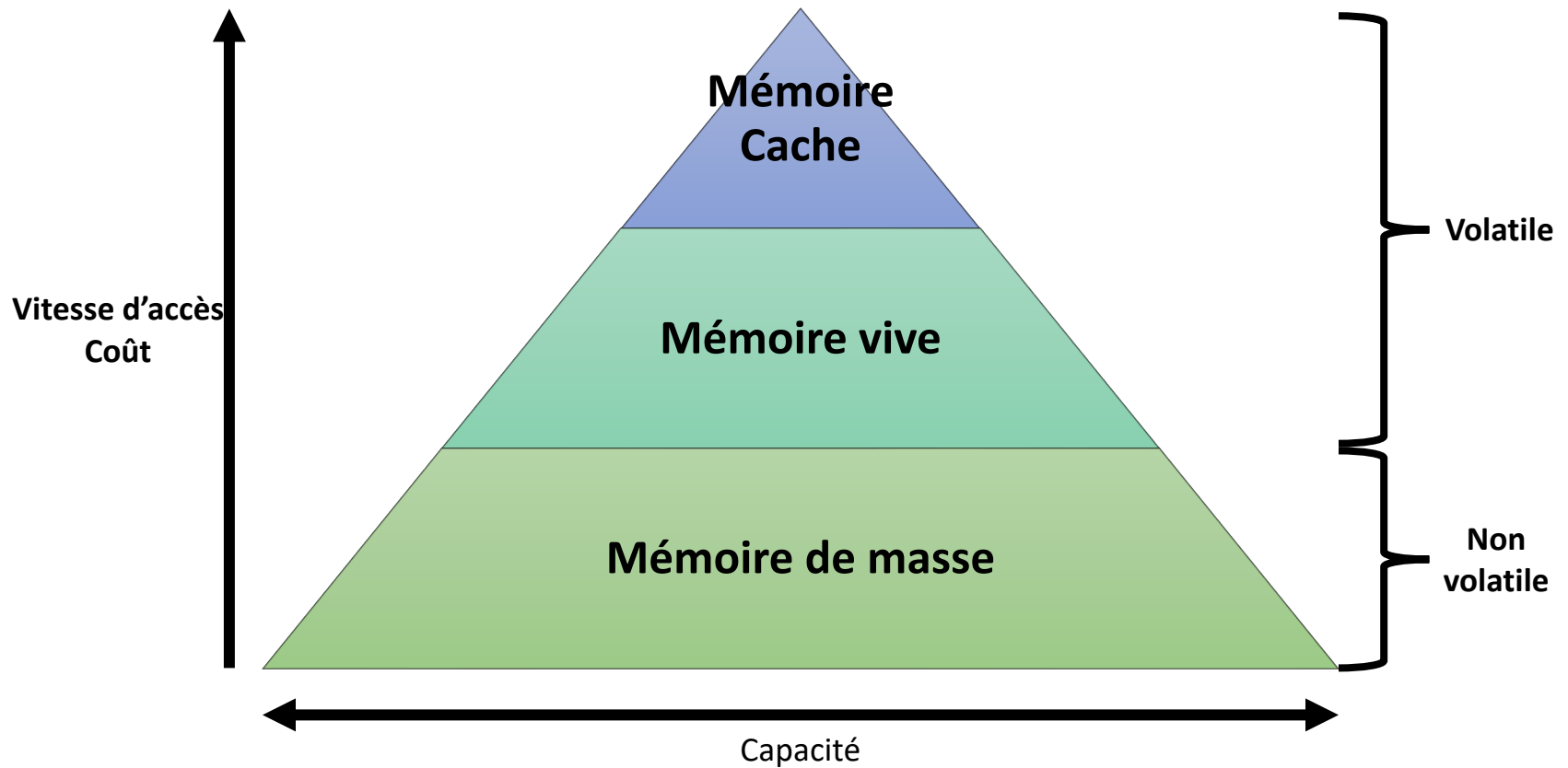
### Chapitre 3

### Mémoire

# La mémoire virtuelle

## Rappels

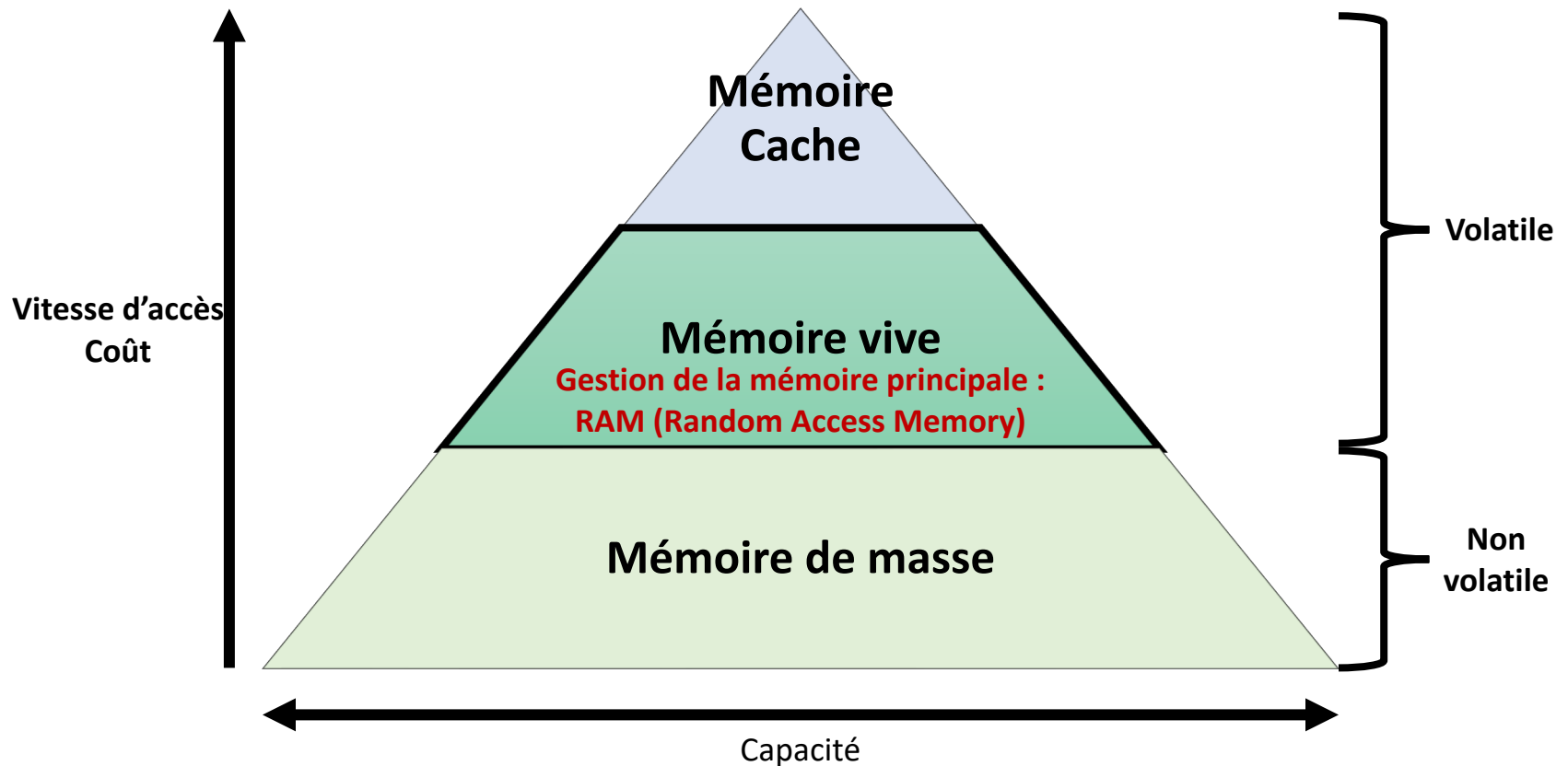
### ● Hiérarchisation de la mémoire



# La mémoire virtuelle

## Rappels

### ● Hiérarchisation de la mémoire

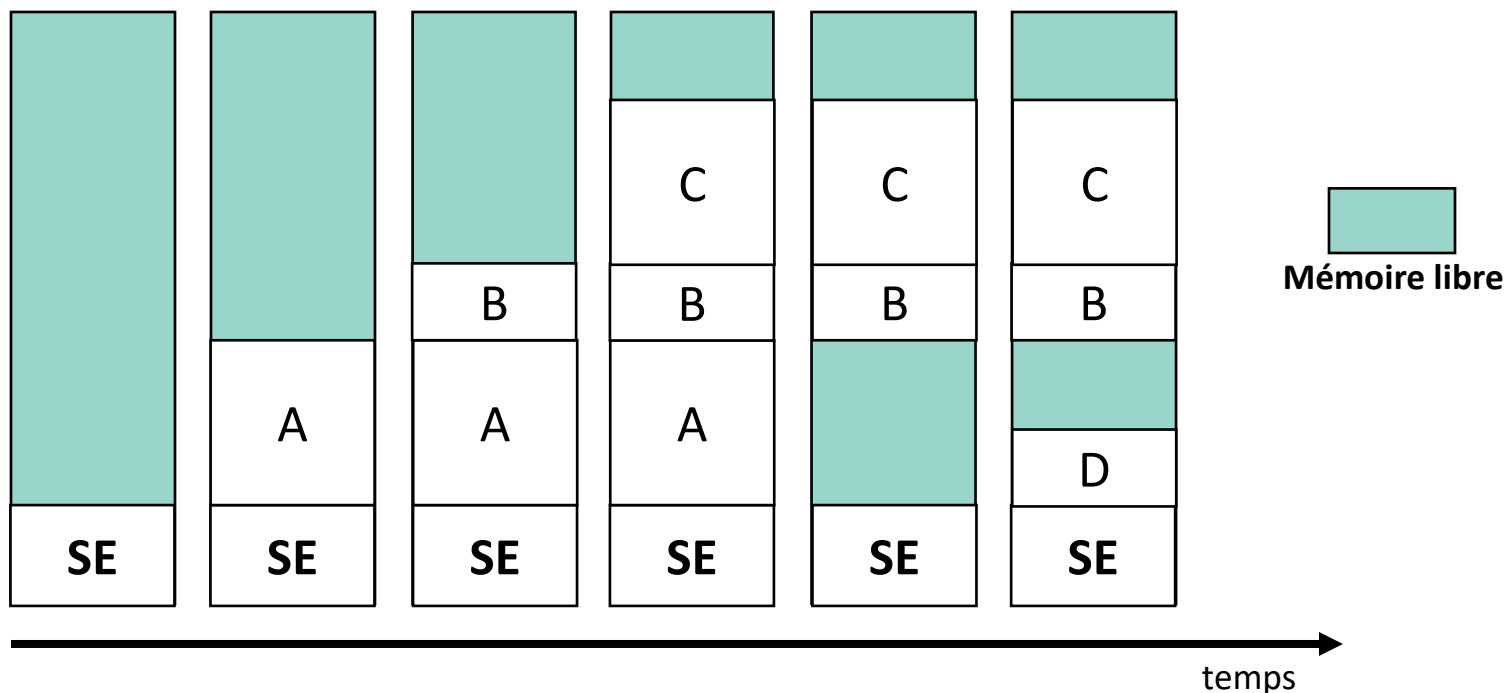


# La gestion de la mémoire

## Rappels

### ● Stratégie va-et-vient

- Définit au chargement, un bloc d'espace mémoire contigüe :



- Lorsque le processus est terminé, il est stocké sur le disque
- **Problème de fragmentation**



Introduction

La gestion de la mémoire

**La mémoire virtuelle**

Les algorithmes de remplacement des pages

La segmentation

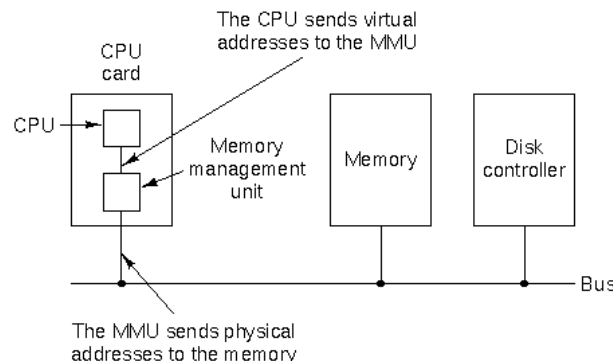
Conclusion

# La mémoire virtuelle

## Introduction

### ● La mémoire virtuelle

- Les programmes génèrent des adresses virtuelles
  - Elles forment l'espace d'adressage virtuel
- Dans un système sans mémoire virtuelle
  - L'adresse est directement placée dans le bus mémoire
  - Provoque la lecture/écriture du mot dans l'adresse physique
- Dans un système avec mémoire virtuelle
  - L'adresse va dans le gestionnaire de mémoire (MMU)
  - Elle fait correspondre les adresses virtuelles et les adresses physiques



# La mémoire virtuelle

## Introduction

### ● La mémoire virtuelle

- Chaque programme a son propre espace d'adressage découpé en petites zones
  - **Les pages**
  - Entités formées d'une suite d'adresses contiguës
- Ces pages sont mappées sur la mémoire physique
  - Il n'est pas obligatoire d'avoir toutes les pages en mémoire physique pour exécuter un programme
- Lorsqu'un programme référence une partie de son espace d'adressage en mémoire physique, le matériel fait la correspondance **à la volée**

# La mémoire virtuelle

## Introduction

### ● La mémoire virtuelle

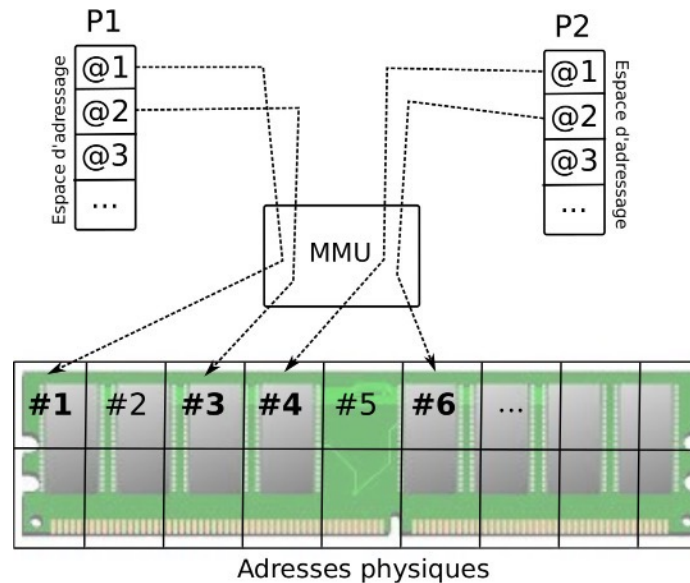
- Si la partie de l'espace référencé n'est pas en mémoire, le SE prend la main :
  - Il va chercher ce qui manque
  - Il le place en mémoire
  - Il reprend là où il s'était arrêté
- Fonctionne très bien dans un système multiprogrammé
  - Les parties de plusieurs programmes sont en mémoire simultanément
  - Lorsqu'un programme attend le chargement en mémoire d'une partie de lui-même, il est en phase **d'attente E/S**
    - L'UC peut choisir un autre processus
- Mécanisme de gestion de la mémoire virtuelle : **la pagination**
  - Utilisé par la plupart des SE



# La mémoire virtuelle

## La pagination

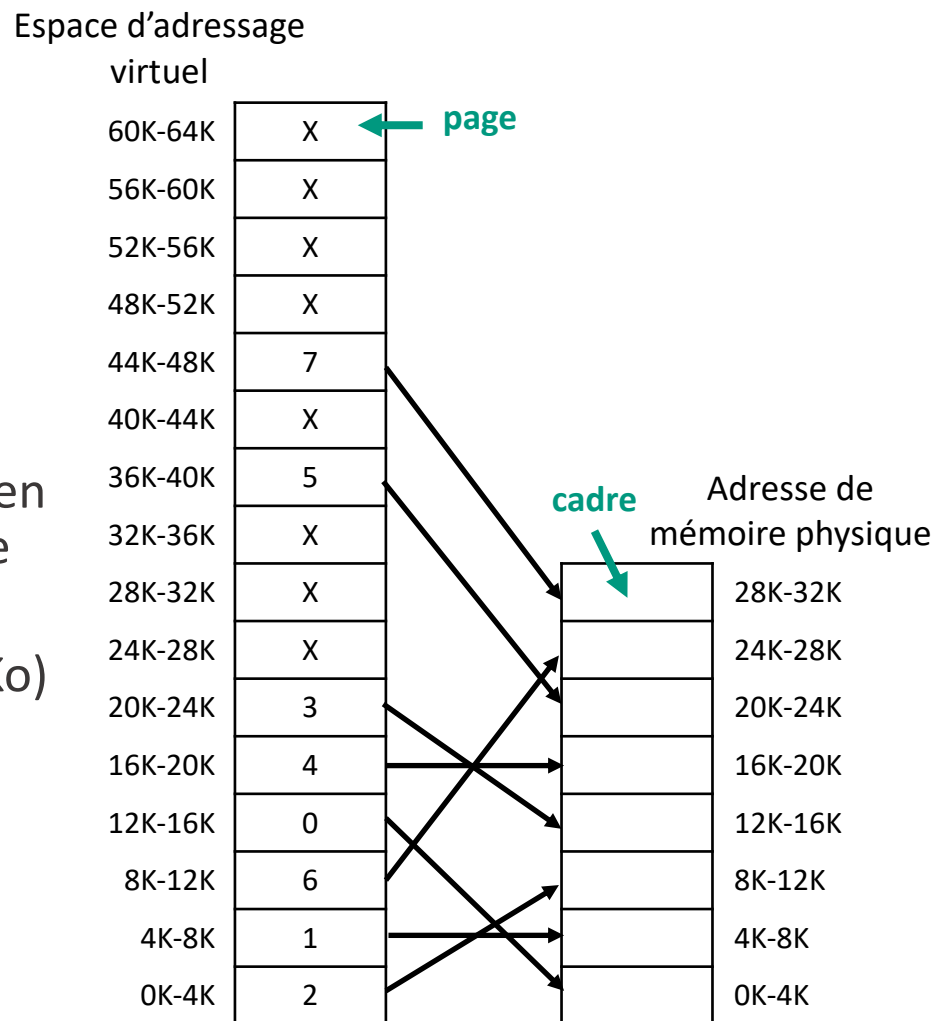
- L'espace d'adressage virtuel est divisé en unités : **les pages**
- La mémoire physique est découpé en unités : **les cadres**
- Les pages et les cadres sont toujours de même taille
  - Variable de 512 octets à 64 KO



# La mémoire virtuelle

## La pagination

- Adresses virtuelles
  - Sur 16 bits
  - Valeurs entre 0 et 64Ko
- 32 Ko de mémoire
  - Possibilité d'écrire des programmes de 64 Ko
  - Ne peuvent être chargés en même temps en mémoire pour exécution
  - Une copie complète (64 Ko) doit être présente sur le disque

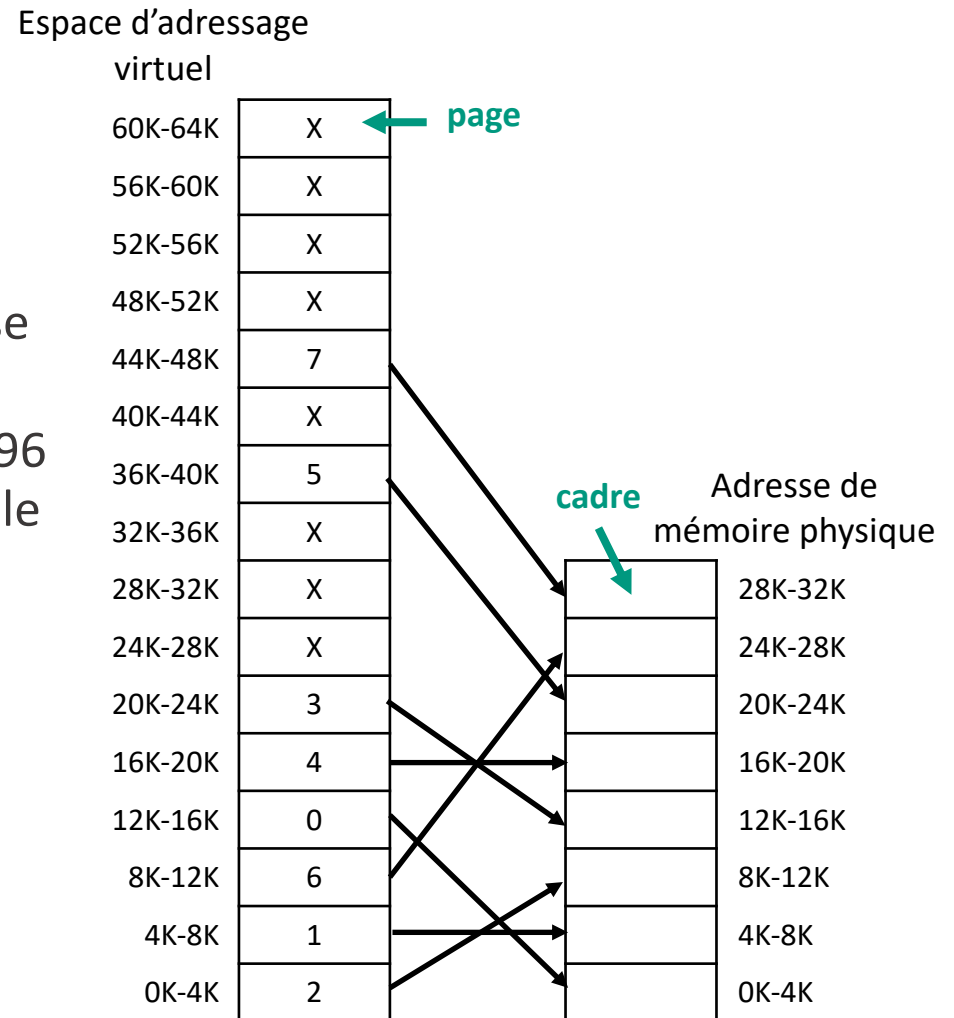


# La mémoire virtuelle

## La pagination

### ● Pages et cadres

- Taille de 4 Ko
  - 16 pages
  - 8 cadres
- 0K-4K signifie que les @ se situent entre 0 et 4095
- Chaque page contient 4096 @ démarrant à un multiple de 4096



# La mémoire virtuelle

## Exemple

### ● Accès adresse 0

- L'@ virtuelle est envoyée au gestionnaire de mémoire

➡ @ 0

Espace d'adressage  
virtuel

60K-64K	X	← page
56K-60K	X	
52K-56K	X	
48K-52K	X	
44K-48K	7	
40K-44K	X	
36K-40K	5	
32K-36K	X	
28K-32K	X	
24K-28K	X	
20K-24K	3	
16K-20K	4	
12K-16K	0	
8K-12K	6	
4K-8K	1	
0K-4K	2	

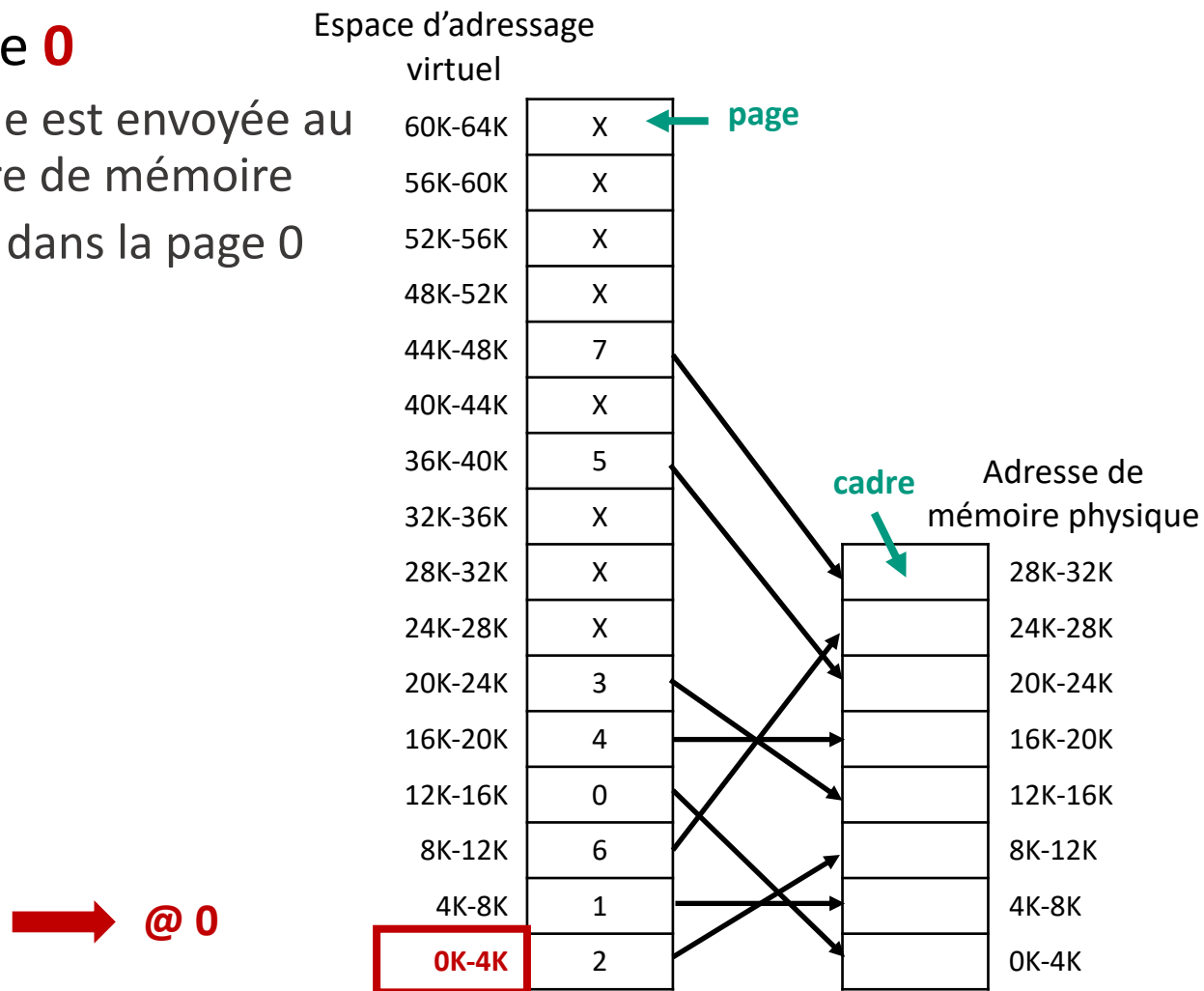
	Adresse de mémoire physique
cadre	28K-32K
	24K-28K
	20K-24K
	16K-20K
	12K-16K
	8K-12K
	4K-8K
	0K-4K

# La mémoire virtuelle

## Exemple

### ● Accès adresse **0**

- L'@ virtuelle est envoyée au gestionnaire de mémoire
- L'@ tombe dans la page 0



# La mémoire virtuelle

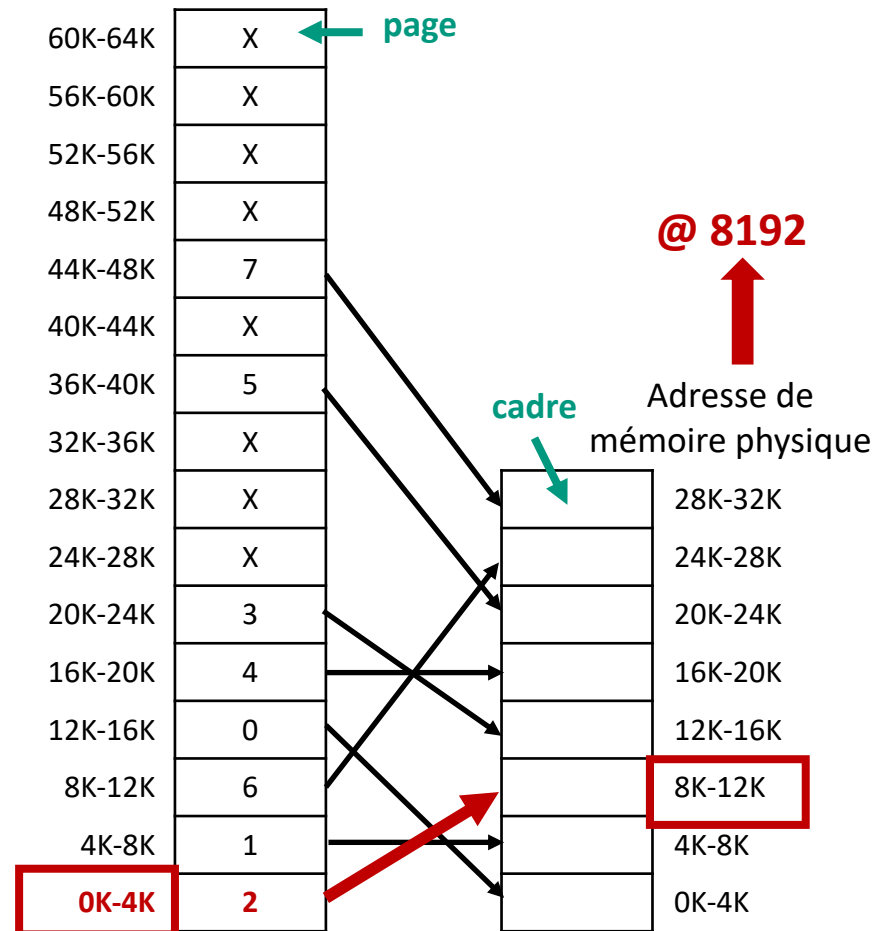
## Exemple

### Accès adresse 0

- L'@ virtuelle est envoyée au gestionnaire de mémoire
- L'@ tombe dans la page 0
- Correspond au cadre 2
  - 8192 – 12287
- L'@ est transformée en **8192** et envoyée sur le bus
- La RAM ne sait rien du MMU
  - Elle interprète la requête avec l'@ 8192

→ @ 0

Espace d'adressage  
virtuel



# La mémoire virtuelle

## Exemple

- Accès adresse **8192**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire

➔ @ 8192

Espace d'adressage  
virtuel

60K-64K	X	← page
56K-60K	X	
52K-56K	X	
48K-52K	X	
44K-48K	7	
40K-44K	X	
36K-40K	5	
32K-36K	X	
28K-32K	X	
24K-28K	X	
20K-24K	3	
16K-20K	4	
12K-16K	0	
8K-12K	6	
4K-8K	1	
0K-4K	2	

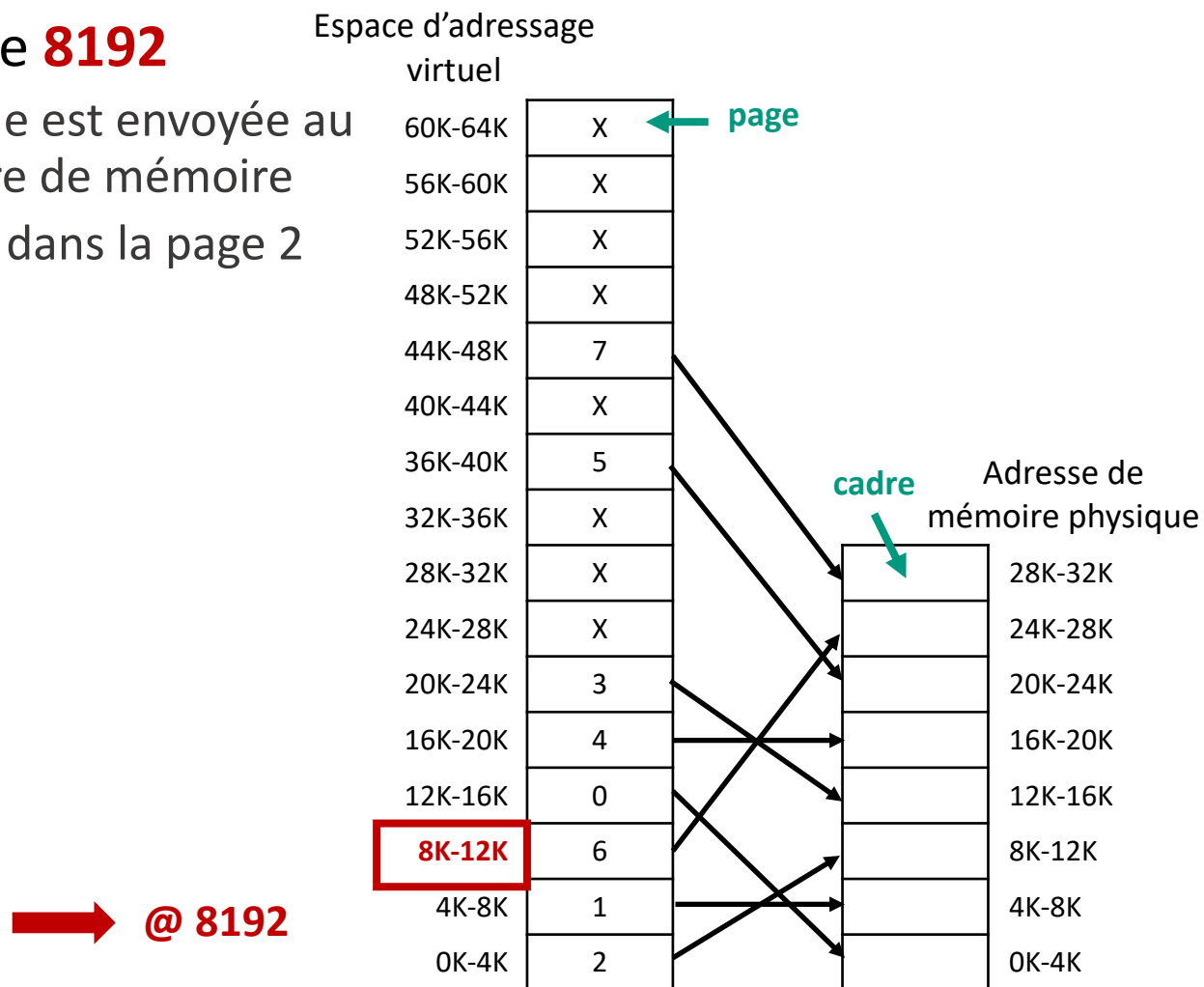
cadre Adresse de  
mémoire physique

28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K

# La mémoire virtuelle

## Exemple

- Accès adresse **8192**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 2

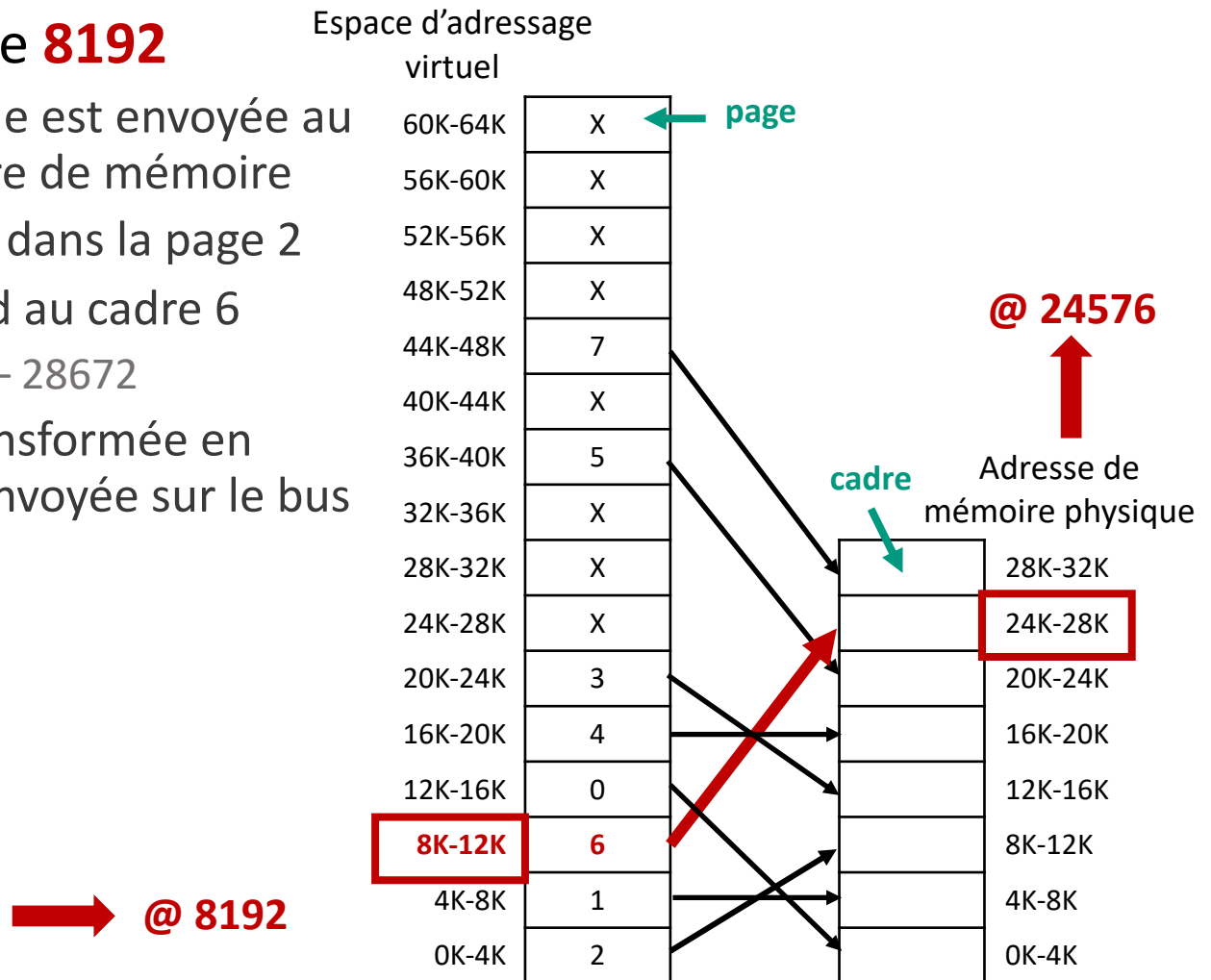




# La mémoire virtuelle

## Exemple

- Accès adresse **8192**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 2
  - Correspond au cadre 6
    - 24576 – 28672
  - L'@ est transformée en **24576** et envoyée sur le bus



# La mémoire virtuelle

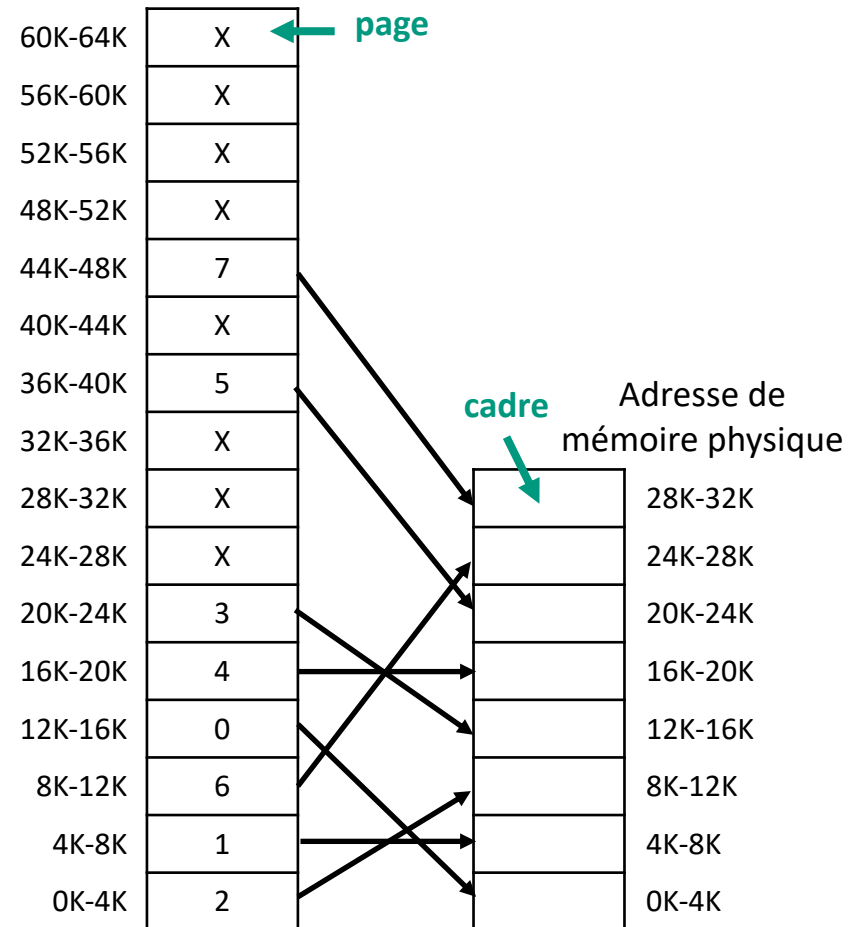
## Exemple

### ● Accès adresse **20500**

- L'@ virtuelle est envoyée au gestionnaire de mémoire

➔ **@ 20500**

Espace d'adressage  
virtuel



# La mémoire virtuelle

## Exemple

### ● Accès adresse **20500**

- L'@ virtuelle est envoyée au gestionnaire de mémoire
- L'@ tombe dans la page 5
  - 20480 – 24576
  - A **20** octets du début

➔ @ 20500

Espace d'adressage  
virtuel

60K-64K	X	← page
56K-60K	X	
52K-56K	X	
48K-52K	X	
44K-48K	7	
40K-44K	X	
36K-40K	5	
32K-36K	X	
28K-32K	X	
24K-28K	X	
<b>20K-24K</b>	3	
16K-20K	4	
12K-16K	0	
8K-12K	6	
4K-8K	1	
0K-4K	2	

	Adresse de mémoire physique
	28K-32K
	24K-28K
	20K-24K
	16K-20K
	12K-16K
	8K-12K
	4K-8K
	0K-4K

# La mémoire virtuelle

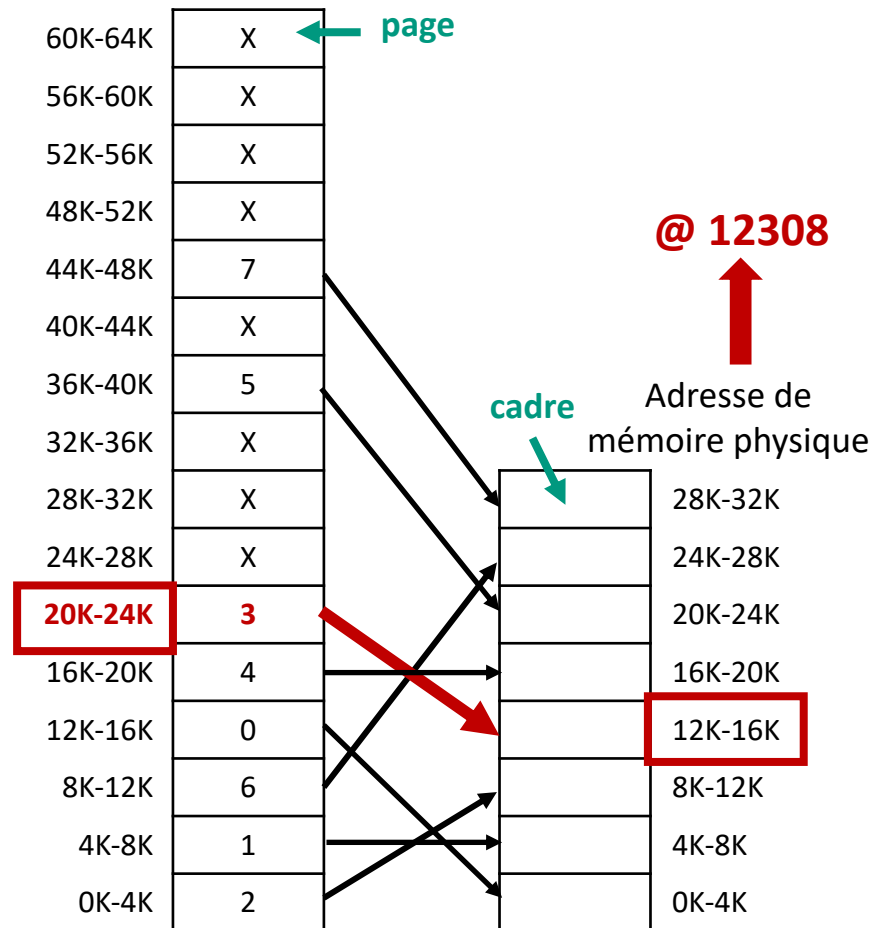
## Exemple

### ● Accès adresse **20500**

- L'@ virtuelle est envoyée au gestionnaire de mémoire
- L'@ tombe dans la page 5
  - 20480 – 24576
  - A **20** octets du début
- Correspond au cadre 3
  - 12288 – 16384
- L'@ est transformée en **12308** et envoyée sur le bus
  - $12288 + 20 = 12308$

➡ @ 20500

Espace d'adressage  
virtuel



@ 12308

Adresse de  
mémoire physique

# La mémoire virtuelle

## Exemple

### ● Accès adresse **32780**

- L'@ virtuelle est envoyée au gestionnaire de mémoire

➔ **@ 32780**

Espace d'adressage  
virtuel

60K-64K	X	← page
56K-60K	X	
52K-56K	X	
48K-52K	X	
44K-48K	7	
40K-44K	X	
36K-40K	5	
32K-36K	X	
28K-32K	X	
24K-28K	X	
20K-24K	3	
16K-20K	4	
12K-16K	0	
8K-12K	6	
4K-8K	1	
0K-4K	2	

	Adresse de mémoire physique
cadre	28K-32K
	24K-28K
	20K-24K
	16K-20K
	12K-16K
	8K-12K
	4K-8K
	0K-4K

# La mémoire virtuelle

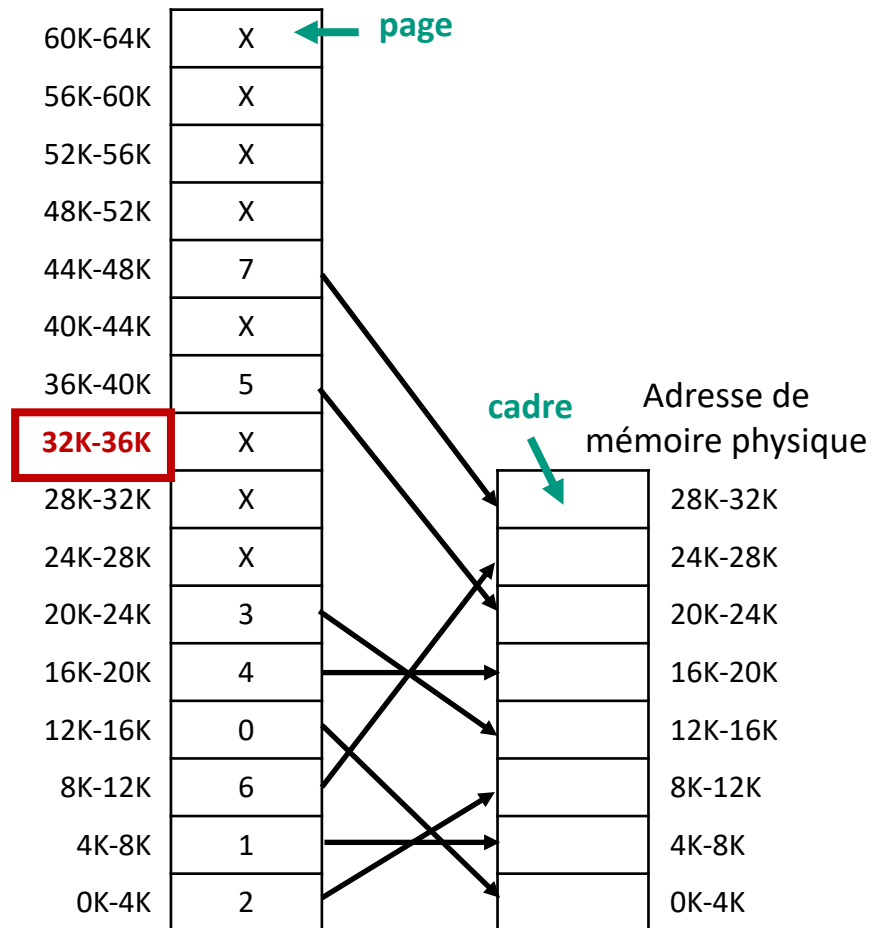
## Exemple

### ● Accès adresse **32780**

- L'@ virtuelle est envoyée au gestionnaire de mémoire
- L'@ tombe dans la page 8
  - 32768 – 36864
  - A **12** octets du début

➡ @ 32780

Espace d'adressage  
virtuel



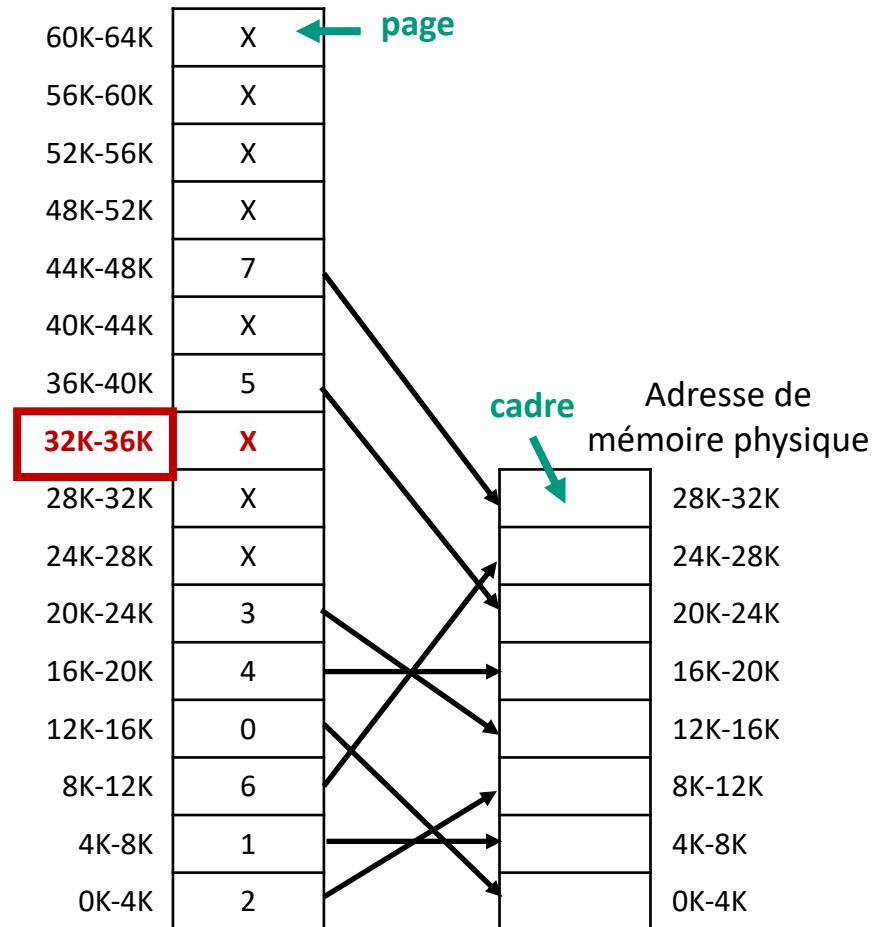
# La mémoire virtuelle

## Exemple

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - **La page est absente**
    - Pas de référence au cadre

➔ @ 32780

Espace d'adressage  
virtuel



# La mémoire virtuelle

## Défaut de page

- Que se passe-t-il si un programme essaye de faire appel à une page non présente ?
  - Le gestionnaire de mémoire remarque que la page est absente
  - Il fait procéder l'UC à un déroutement :
    - Le SE sélectionne un cadre peu utilisé
    - Il écrit son contenu sur le disque
    - Il transfère la page qui vient d'être référencée dans le cadre libéré
    - Il modifie la correspondance
    - Il recommence l'instruction déroutée
  - C'est le **défaut de page**



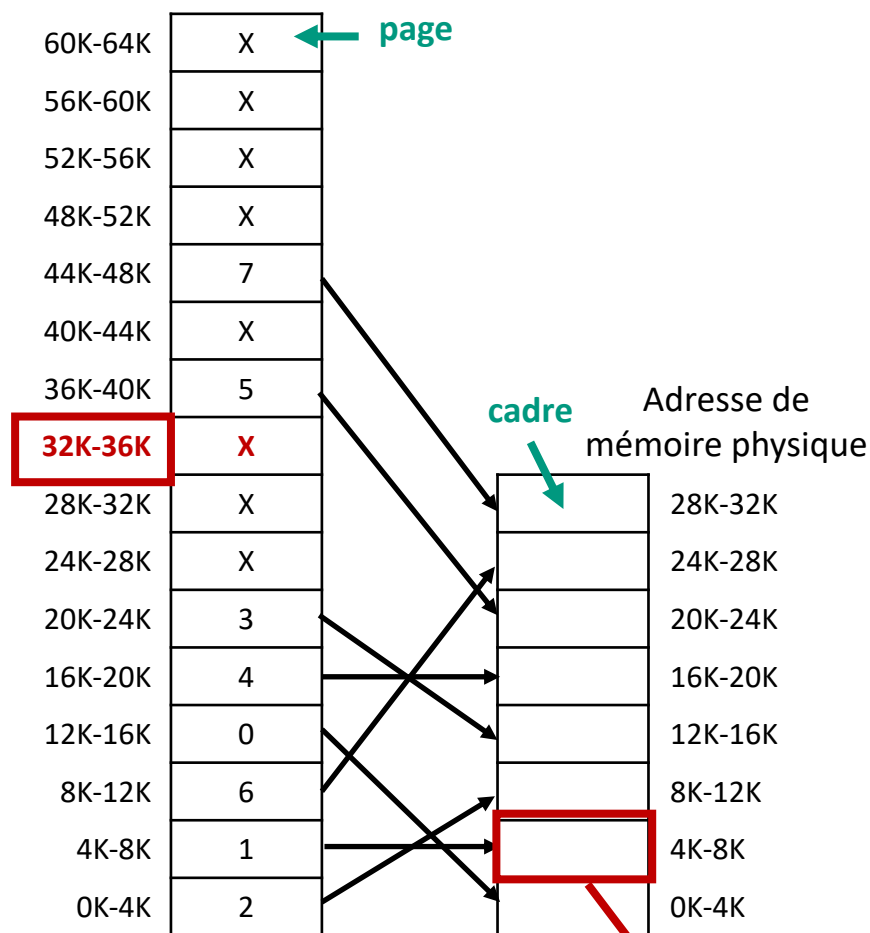
# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - La page est absente
    - Copie cadre 1 sur le disque

➡ @ 32780

Espace d'adressage  
virtuel



Copie sur disque

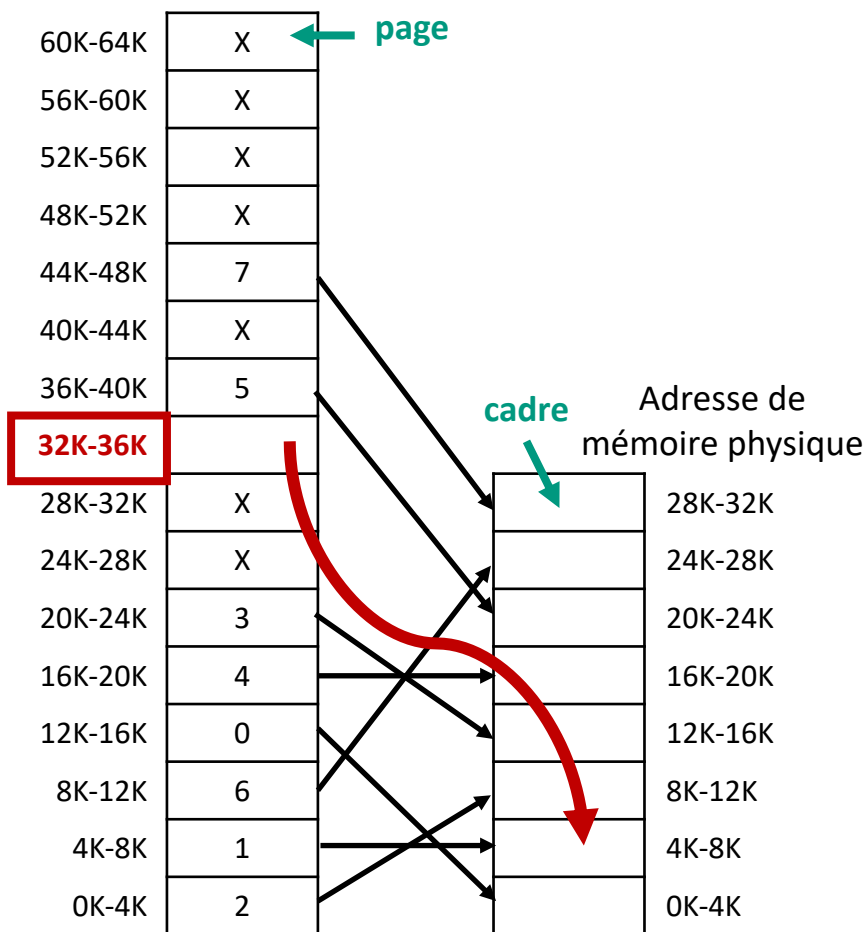
# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - La page est absente
    - Copie cadre 1 sur le disque
    - Charge page 8 vers cadre 1

➔ @ 32780

Espace d'adressage  
virtuel



# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
- La page est absente
  - Copie cadre 1 sur le disque
  - Charge page 8 vers cadre 1
  - Modifie les correspondances
- Reprise de l'instruction
  - @ **4108** utilisée
    - $4096 + 12 = 4108$

➡ @ 32780

Espace d'adressage  
virtuel

60K-64K	X	← page
56K-60K	X	
52K-56K	X	
48K-52K	X	
44K-48K	7	
40K-44K	X	
36K-40K	5	
<b>32K-36K</b>	<b>1</b>	
28K-32K	X	
24K-28K	X	
20K-24K	3	
16K-20K	4	
12K-16K	0	
8K-12K	6	
4K-8K	X	
0K-4K	2	

cadre

Adresse de  
mémoire physique

@ 4108




28K-32K
24K-28K
20K-24K
16K-20K
12K-16K
8K-12K
4K-8K
0K-4K

# La mémoire virtuelle

## La pagination

- Conversion des @ virtuelles en @ physique
  - Le rôle du gestionnaire de mémoire
    - Memory Management Unit (MMU)
  - Une adresse virtuelle est un couple (numéro de page, déplacement)
    - Numéro de page = adresse / taille des pages (division entière)
    - Déplacement = adresse % taille des pages (modulo)
    - Taille des pages variable selon les systèmes
  - Ces deux opérations sont réalisées très efficacement lorsque la taille des pages est une puissance de 2
  - Exemple :

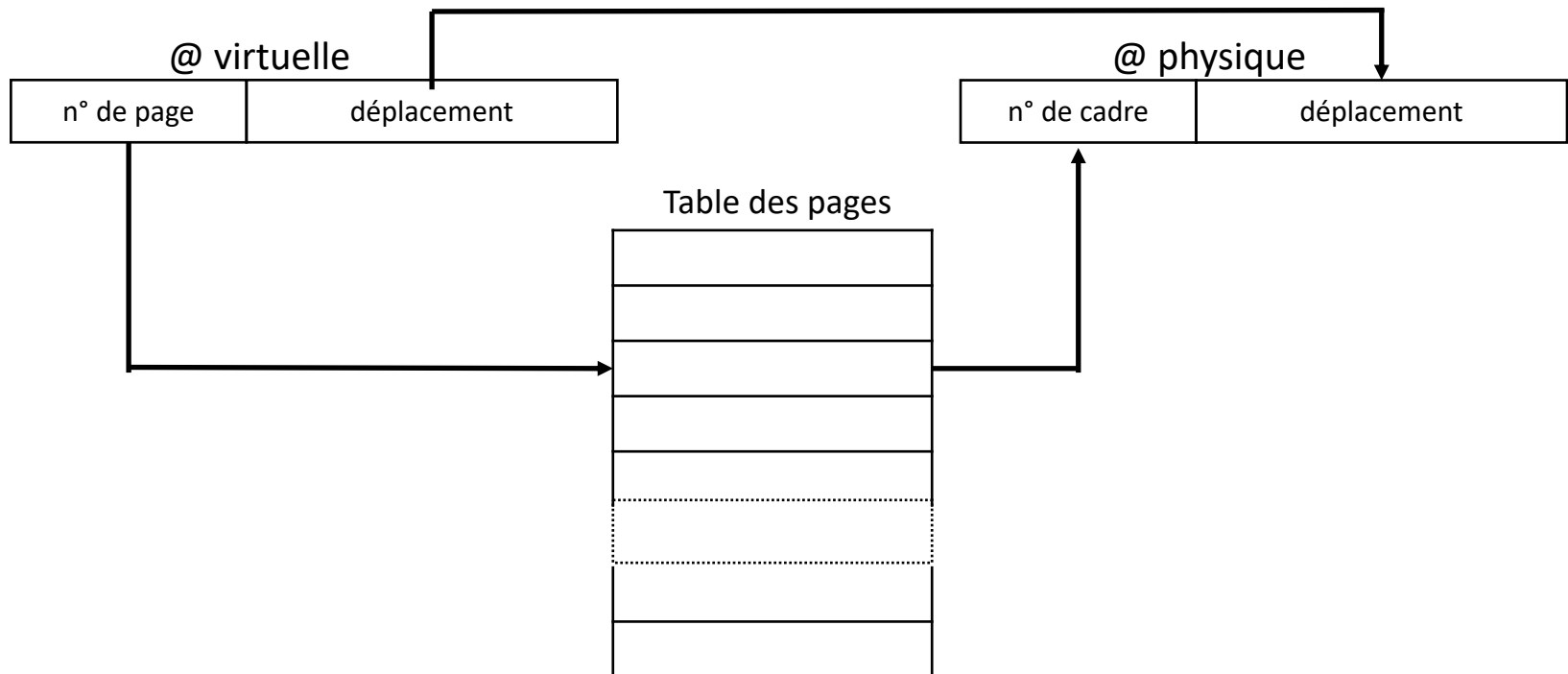
- Adresse virtuelle sur 16 bits avec numéro de page sur 4 bits

00100000000000100  0010 0000000000100  
Numéro de page : 2 Déplacement : 4

# La mémoire virtuelle

## La pagination

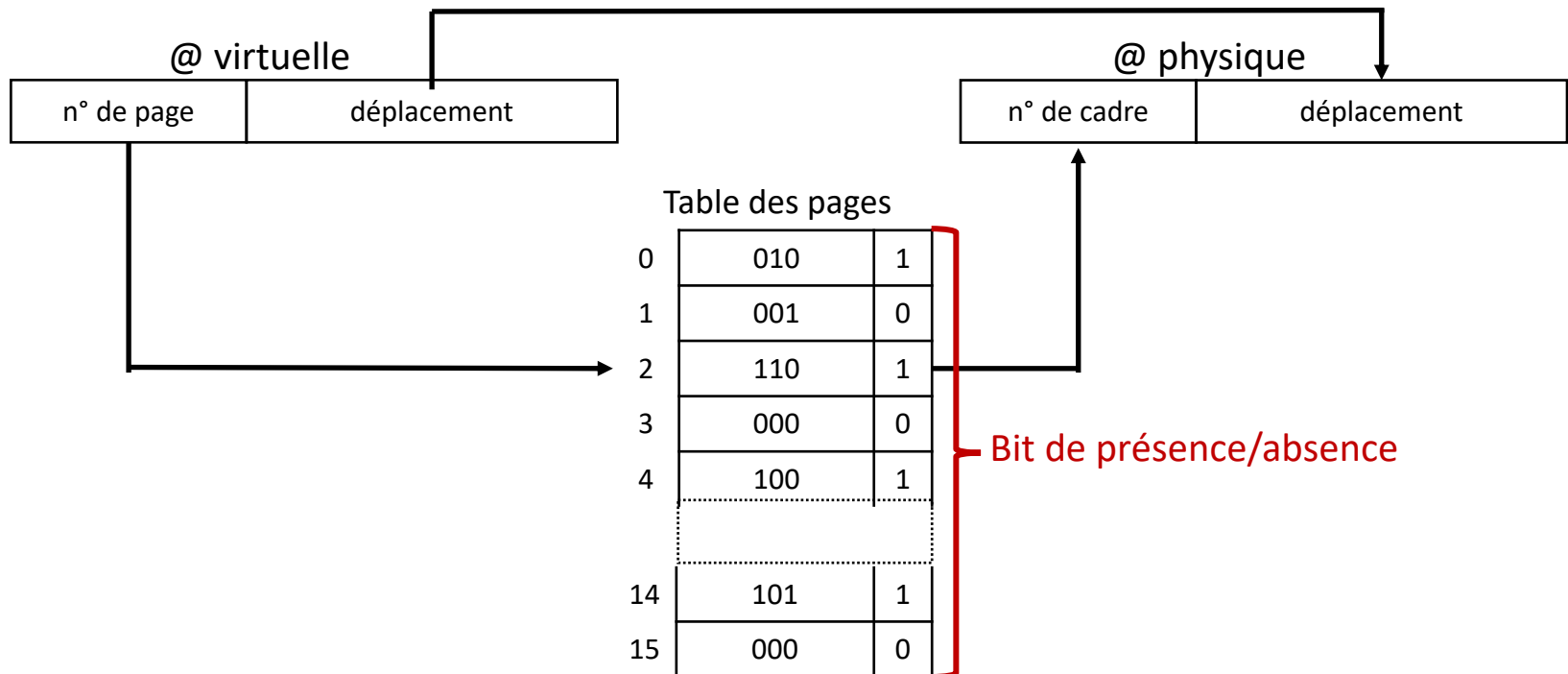
- Conversion des @ virtuelles en @ physique
  - Utilisation d'une table des pages
    - Correspondance n° de page  $\leftrightarrow$  n° de cadre
    - Le déplacement est le même



# La mémoire virtuelle

## La pagination

- Conversion des @ virtuelles en @ physique
  - Utilisation d'une table des pages
    - Correspondance n° de page <-> n° de cadre
    - Le déplacement est le même



# La mémoire virtuelle

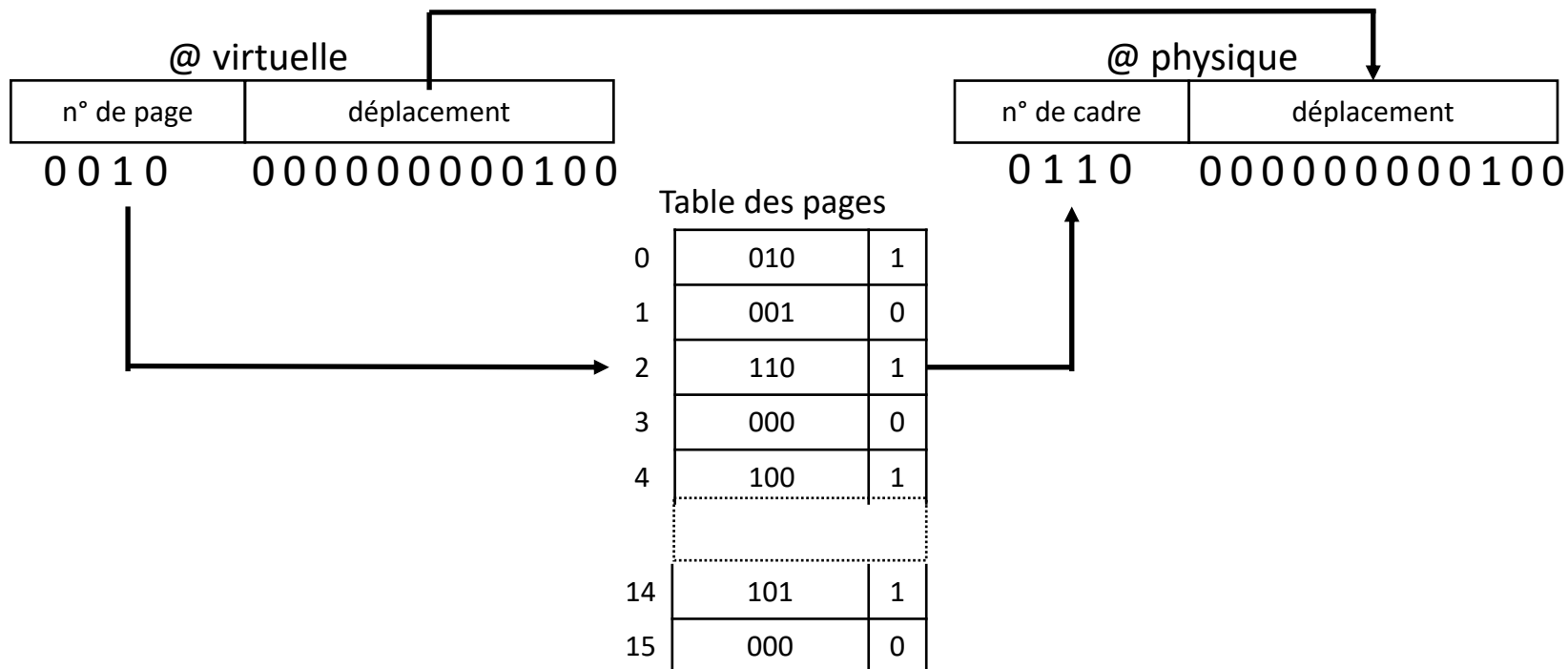
## La pagination

### Conversion des @ virtuelles en @ physique

#### Utilisation d'une table des pages

Correspondance n° de page <-> n° de cadre

Le déplacement est le même



# La mémoire virtuelle

## La pagination

### Question

- Les adresses sont sur 16 bits
- L'espace d'adressage d'un processus est de  $2^{16} = 64$  Ko
- Le MMU impose une taille de cadre et de page de 4 Ko ( $2^{12}$ )
- Le nombre max de pages pour un processus =  $2^{16}/2^{12} = 16$

- Dans quel cadre se trouve l'adresse virtuelle 20500 ?

- A quelle adresse physique correspond-elle ?

page	cadre
0	010
1	001
2	110
3	000
4	100
5	011
6	111
...	...
14	101
15	000



# La mémoire virtuelle

## La pagination

### Exercice

20500 =  $2^{14} + 2^{12} + 2^4 + 2^2$



0101

Numéro  
de page :  
**5**

000000010100

Déplacement :  
**20**

### @ physique ?

Binaire 0011 000000010100

**@ 12308**

page	cadre
0	010
1	001
2	110
3	000
4	100
5	011
6	111
...	...
14	101
15	000



Introduction

La gestion de la mémoire

La mémoire virtuelle

**Les algorithmes de remplacement des pages**

La segmentation

Conclusion

# Les algorithmes de remplacement des pages

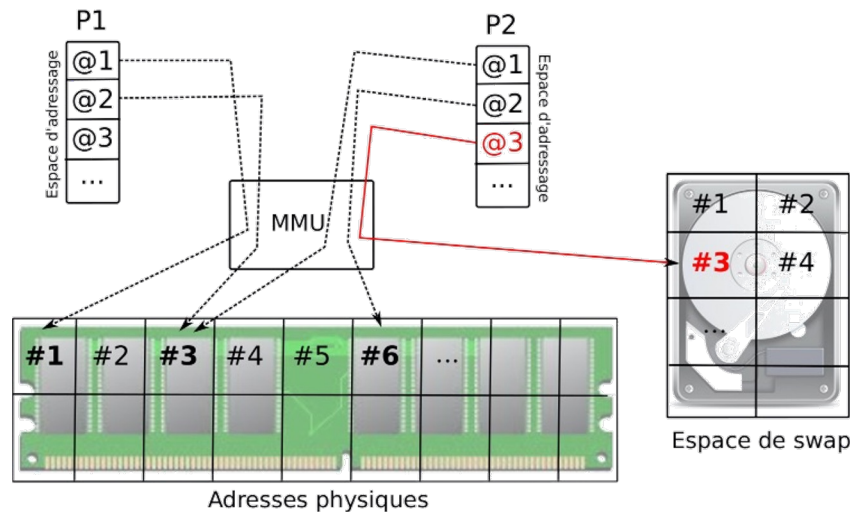
## Le swapping

### ● Rappel

- Besoin de placer un cadre en RAM sur le disque dur

### ● Le **swapping**

- Consiste à utiliser une mémoire secondaire (disque dur) comme extension de la RAM
- La zone de swap est une zone particulière du disque dur
  - Contient des couples (identifiant de cadre, contenu)



# Les algorithmes de remplacement des pages

## Le dirty bit

### ● Rappel

- Lorsqu'une page est évincée de la mémoire pour être stockée dans le swap, il peut arriver que le contenu de cette page soit déjà dans le swap
- Un bit spécial dans l'entrée de la table des pages est utilisé
  - Le dirty bit
- Eviter de recopier inutilement le contenu de la page

### ● Une page désignée par une adresse virtuelle peut être dans 3 états différents :

- **Utilisée** : actuellement associée à un cadre en RAM
- **Invalide** : ne correspond à aucun cadre
- **Swappée** : le cadre correspondant est actuellement swappé en mémoire secondaire

# Les algorithmes de remplacement des pages

## Défaut de page

- **Pagination à la demande**
  - L'état de la page est vérifié à chaque accès mémoire
  - Si la page est swapée, une interruption appelée **défaut de page** donne la main au SE
- **Le SE réalise alors les tâches suivantes :**
  - Trouver un cadre
    - Prendre un cadre libre s'il reste de la mémoire
    - Choisir un cadre occupé et le swapper si la mémoire est pleine
  - Copier le contenu de la page depuis la mémoire secondaire vers le cadre trouvé et mettre à jour la table des pages du processus
  - Reprendre l'exécution du processus à l'instruction qui a provoqué le défaut de page

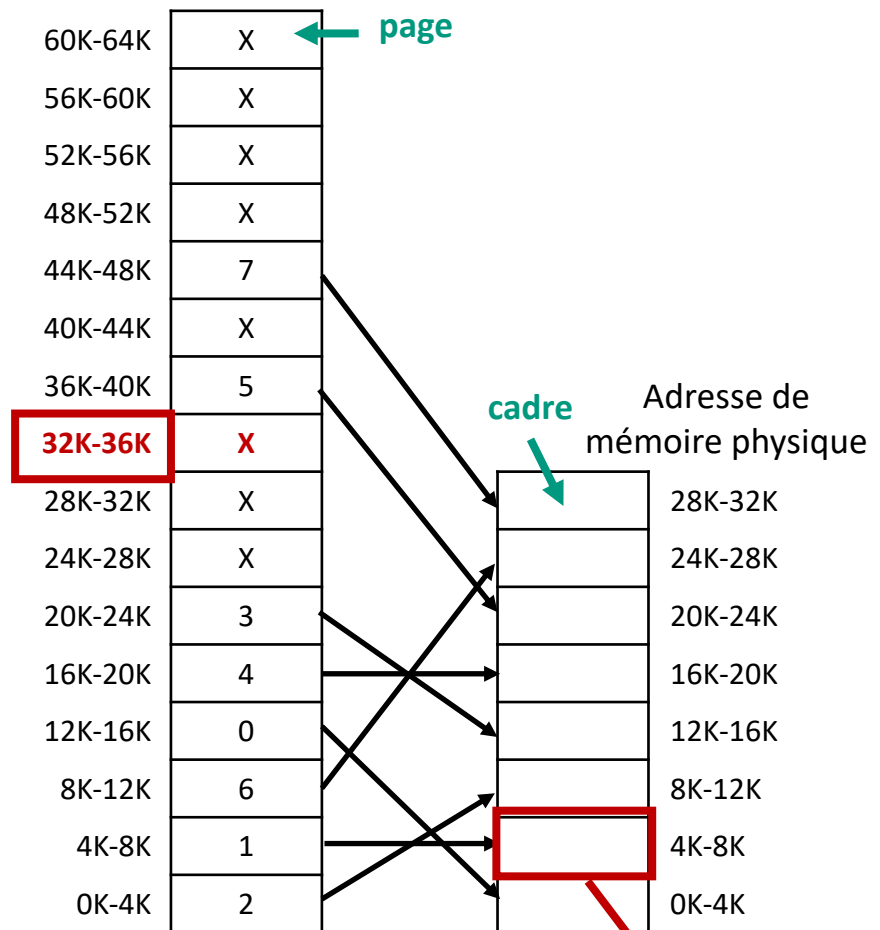
# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - La page est absente
    - Copie cadre 1 sur le disque

➡ @ 32780

Espace d'adressage  
virtuel



Copie sur disque

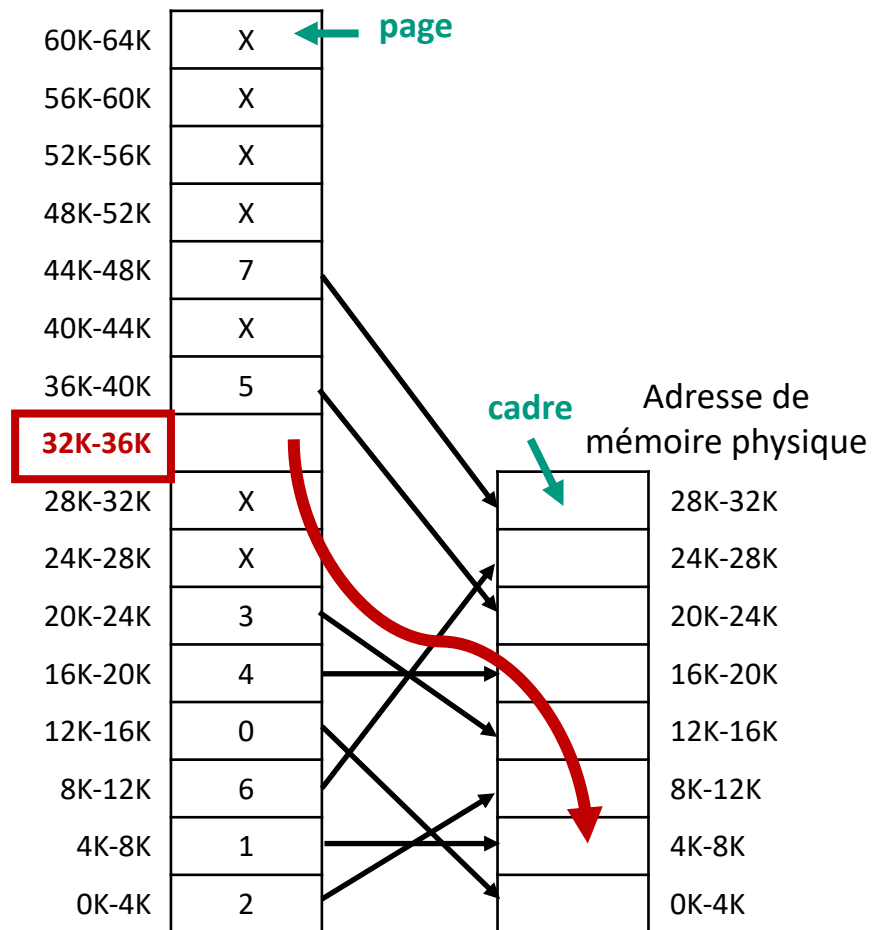
# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - La page est absente
    - Copie cadre 1 sur le disque
    - Charge page 8 vers cadre 1

➡ @ 32780

Espace d'adressage  
virtuel



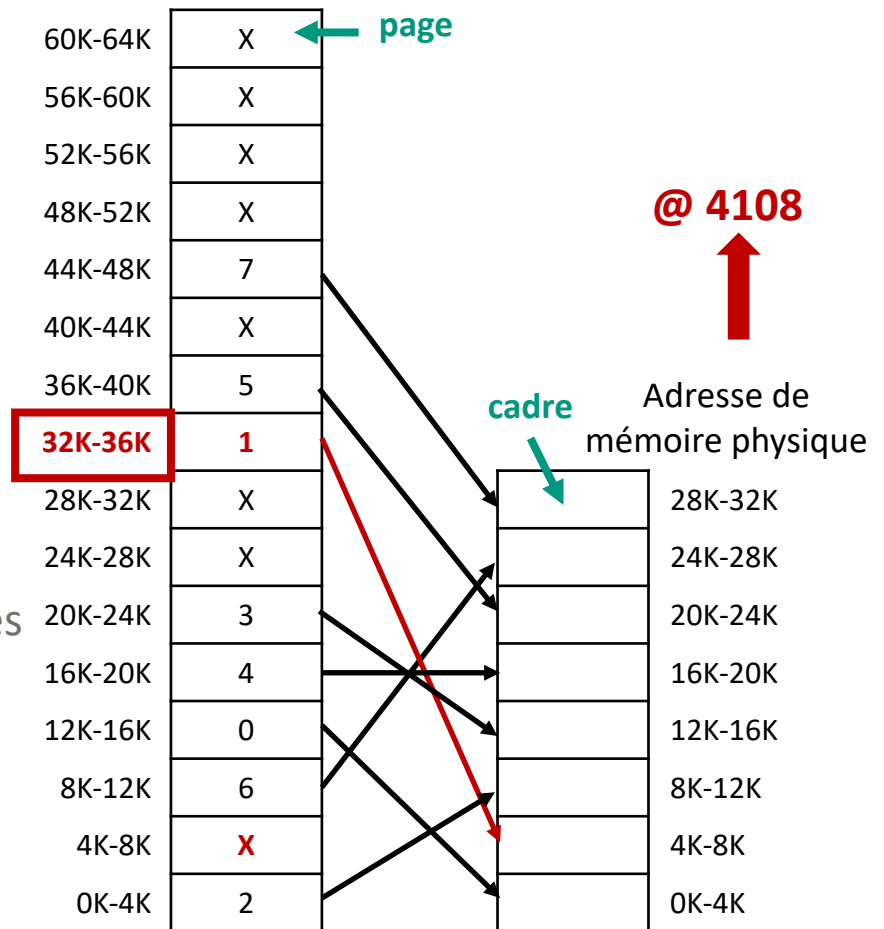
# La mémoire virtuelle

## Exemple défaut de page

- Accès adresse **32780**
  - L'@ virtuelle est envoyée au gestionnaire de mémoire
  - L'@ tombe dans la page 8
    - 32768 – 36864
    - A **12** octets du début
  - La page est absente
    - Copie cadre 1 sur le disque
    - Charge page 8 vers cadre 1
    - Modifie les correspondances
  - Reprise de l'instruction
    - @ **4108** utilisée
      - $4096 + 12 = 4108$

➡ @ 32780

Espace d'adressage  
virtuel





# Les algorithmes de remplacement des pages

## Problématique

- ① Les défauts de pages sont coûteux
  - ① Temps d'accès au disque
  - ① Ils sont traités de manière prioritaire par le SE
- ① Lorsque la mémoire est pleine ?
  - ① Le choix de la page qui sera swappée pour faire de la place à la suivante est crucial pour les performances du SE
  - ① Le but est de minimiser le nombre de défauts de pages
- ① Il existe de nombreuses stratégies de remplacement de pages

# Les algorithmes de remplacement des pages

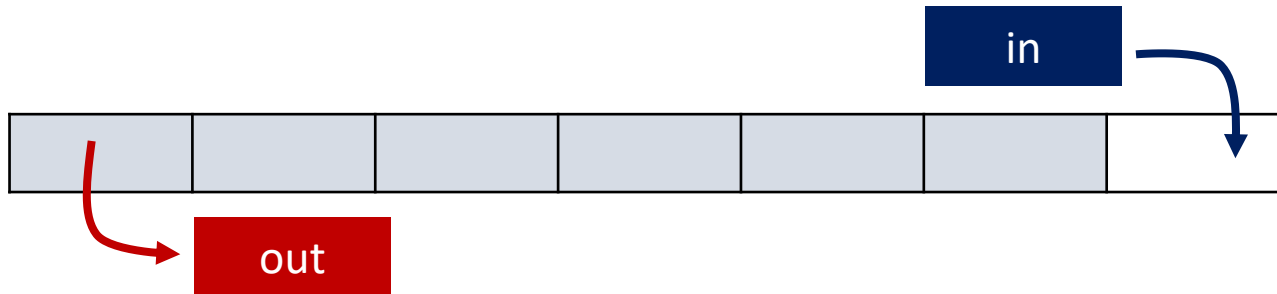
## Algorithme optimal

- La page qui sera utilisée le plus tard dans le futur est évincée
  - C'est la meilleure stratégie
- Mais elle est impossible à implémenter
  - Lorsqu'un défaut de page intervient le SE n'a aucun moyen de savoir quand chacune de ces pages sera référencée la prochaine fois
- On s'en sert comme référence
  - On peut l'implémenter à posteriori dans une deuxième exécution
  - Les algorithmes proposés sont comparés à l'optimal et ont pour but de s'en rapprocher

# Les algorithmes de remplacement des pages

## Algorithme Premier entré, premier sorti (FIFO)

- La page la plus anciennement chargée est évincée
  - L'implémentation concrète demande soit :
    - De conserver la date de chargement de chaque page
      - Permet de choisir la page la plus ancienne
    - De gérer une liste FIFO des pages chargées
      - Lorsqu'une nouvelle page est chargée, elle est mise à la fin
      - Lors d'un défaut de page, la première page de la file est choisie



- Variation : la seconde chance
  - Vérification si la page la plus ancienne n'est pas utilisée

# Les algorithmes de remplacement des pages

## Remplacement de la page la moins récemment utilisée

- Least Recently Used (LRU)
- La page la moins récemment utilisée est évincée
  - Obéit au principe de localité temporelle
  - L'implémentation concrète demande :
    - De gérer une liste FIFO des pages accédées
    - Lorsqu'une nouvelle page est accédée, elle est mise à la fin
    - Lorsqu'il faut libérer un cadre, on prend celui correspondant à la première page de la file
- Problème de surcoût important dû aux réorganisations de la file

# Les algorithmes de remplacement des pages

## Remplacement de la page non récemment utilisée

- Not Recently Used (NRU)
- Une page non référencée et non modifiée est évincée
  - Utilisation du bit de référencé R et modifié M de la table des pages
    - R est mis à 1 quand la page est référencée (lue ou écrite)
    - M est mis à 1 quand la page est modifiée
  - Permet de définir 4 classes
    - Classe 0 : Non référencée, non modifiée
    - Classe 1 : Référencée, non modifiée
    - Classe 2 : non référencée, modifiée
    - Classe 3 : Référencée, modifiée
  - Lors d'un défaut de page, les pages sont classées
    - Une page de la classe la plus basse est choisie au hasard
  - Simple à implémenter
  - Performances correctes

# Les algorithmes de remplacement des pages

## Remplacement de la page la moins fréquemment utilisée

- Least Frequently Used (LFU)
- La page la moins fréquemment utilisée est évincée
  - Obéit au principe de localité temporelle
  - Fréquence implique une période de temps pour la mesure
    - Interruption d'horloge (similairement à l'ordonnancement de type tourniquet)
  - A chaque interruption d'horloge, le SE examine toutes les pages
    - Si le bit R est à 1, on incrémente de compteur de la page
  - Le compteur est décrémenté à intervalles de temps réguliers
    - Divisé par 2 : décalage de bits
    - Permet qu'une page beaucoup utilisée il y a longtemps prenne moins d'importance
  - Lors d'un défaut de page, la page qui a la plus petite fréquence est évincée

# Les algorithmes de remplacement des pages

## Exemples

### ● Exemple 1

- Le tableau ci-dessous présente pour 4 pages, le moment du chargement, le temps du dernier accès, et les bits R et M

Page	Chargement	Dernière référence	R	M
0	t = 126	t = 265	0	0
1	t = 230	t = 280	0	1
2	t = 140	t = 270	0	0
3	t = 110	t = 285	1	1

- Quelle page sera remplacée avec :
  - L'algorithme FIFO ?
  - L'algorithme NRU ?
  - L'algorithme LRU ?
  - L'algorithme de la seconde chance ?

# Les algorithmes de remplacement des pages

## Exemples

### ● Exemple 1

- Le tableau ci-dessous présente pour 4 pages, le moment du chargement, le temps du dernier accès, et les bits R et M

Page	Chargement	Dernière référence	R	M
0	t = 126	t = 265	0	0
1	t = 230	t = 280	0	1
2	t = 140	t = 270	0	0
3	t = 110	t = 285	1	1

- Quelle page sera remplacée avec :
  - L'algorithme FIFO ? **La page 3**
  - L'algorithme NRU ? **La page 0 ou 2**
  - L'algorithme LRU ? **La page 0**
  - L'algorithme de la seconde chance ? **La page 0**



# Les algorithmes de remplacement des pages

## Exemples

### ● Exemple 2

- On considère un système possédant  $N=4$  cadres et  $M=8$  pages
- On part d'une mémoire vide
- On considère un programme générant les accès aux pages dans l'ordre suivant :
  - 1, 5, 2, 6, 8, 2, 5, 6, 3, 7, 6, 4, 5, 4
- Mettez en œuvre les stratégies suivantes et comptez le nombre de défaut de pages
  - L'algorithme FIFO ?
  - L'algorithme LRU ?
  - L'algorithme LFU ?

# Les algorithmes de remplacement des pages

## Exemples

### ● First In First Out (FIFO)

Ordre : 1, 5, 2, 6, 8, 2, 5, 6, 3, 7, 6, 4, 5, 4

Ordre	Cadres			
1				
5				
2				
6				
8				
2				
5				
6				
3				
7				
6				
4				
5				
4				

# Les algorithmes de remplacement des pages

## Exemples

### First In First Out (FIFO)

Ordre : 1, 5, 2, 6, 8, 2, 5, 6, 3, 7, 6, 4, 5, 4

Ordre	Cadres			
1	1			
5	1	5		
2	1	5	2	
6	1	5	2	6
8	8	5	2	6
2	8	5	2	6
5	8	5	2	6
6	8	5	2	6
3	8	3	2	6
7	8	3	7	6
6	8	3	7	6
4	8	3	7	4
5	5	3	7	4
4	5	3	7	4

9 défauts de page

# Les algorithmes de remplacement des pages

## Exemples

### Least Recently Used (LRU)

Ordre : 1, 5, 2, 6, 8, 2, 5, 6, 3, 7, 6, 4, 5, 4

Ordre	Cadres				
1	1				←
5	1	5			←
2	1	5	2		←
6	1	5	2	6	←
8	8	5	2	6	←
2	8	5	2	6	
5	8	5	2	6	
6	8	5	2	6	
3	3	5	2	6	←
7	3	5	7	6	←
6	3	5	7	6	
4	3	4	7	6	←
5	5	4	7	6	←
4	5	4	7	6	

9 défauts de page

# Les algorithmes de remplacement des pages

## Exemples

### Least Frequently Used (LFU)

Ordre : 1, 5, 2, 6, 8, 2, 5, 6, 3, 7, 6, 4, 5, 4

Ordre	Cadres			
1	1			
5	1	5		
2	1	5	2	
6	1	5	2	6
8	8	5	2	6
2	8	5	2	6
5	8	5	2	6
6	8	5	2	6
3	3	5	2	6
7	7	5	2	6
6	7	5	2	6
4	4	5	2	6
5	4	5	2	6
4	4	5	2	6

8 défauts de page

# Les algorithmes de remplacement des pages

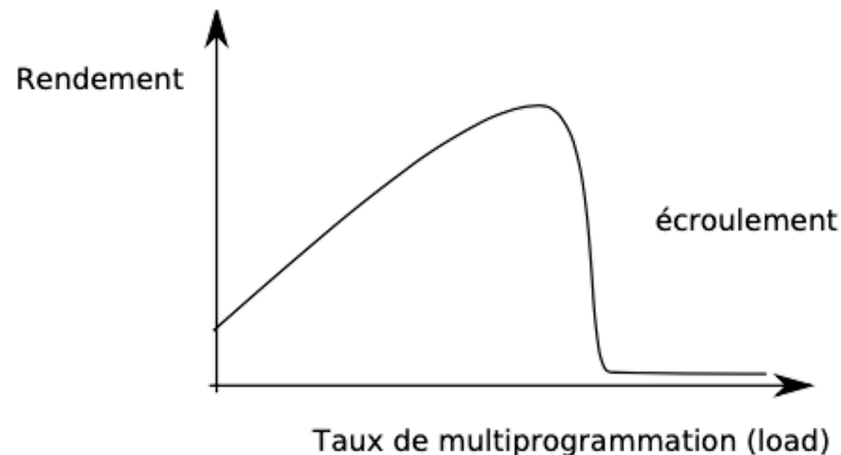
## Le swapping et la multiprogrammation

- Le SE assure une équité du partage de la mémoire entre les processus
  - Le SE dispose de M cadres mémoires qui doivent être partagés entre lui-même et les processus
  - Lorsqu'un remplacement de cadre est nécessaire pour le processus P1
    - Le cadre évincé est toujours choisi parmi les cadres du processus P1
    - C'est une stratégie locale
- Deux types de partition
  - Partition fixe
    - Chaque processus dispose d'un nombre fixe de cadres
  - Partition variable
    - Chaque processus dispose d'un nombre variable de cadres

# Les algorithmes de remplacement des pages

## Le swapping et la multiprogrammation

- Equilibre entre nombre de processus et nombre de cadres
  - Lorsqu'il y a trop de processus par rapport au nombre de cadres
    - Le système passe plus de temps à gérer les défaut de pages
    - Le rendement s'écroule





Introduction

La gestion de la mémoire

La mémoire virtuelle

Les algorithmes de remplacement des pages

**La segmentation**

Conclusion



# La segmentation

## Introduction

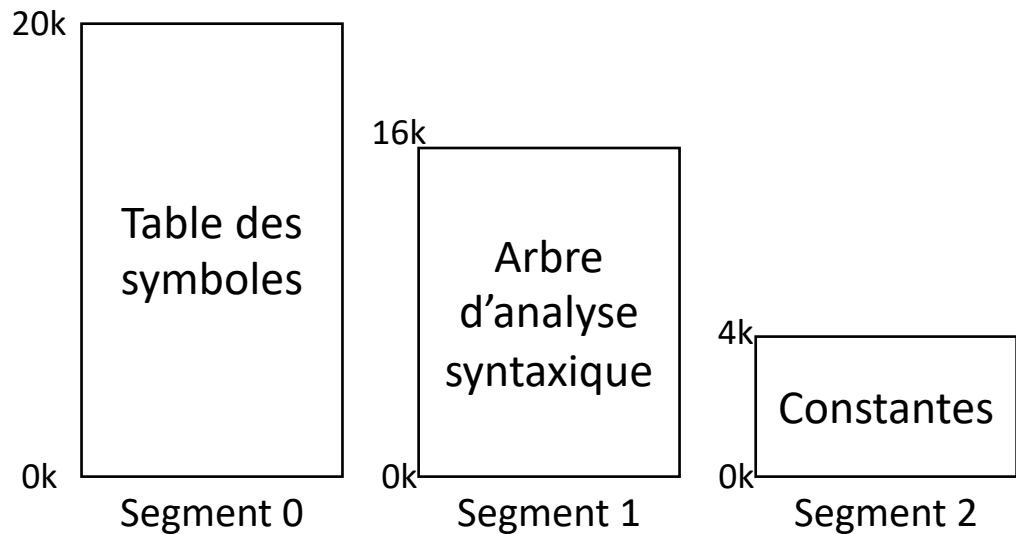
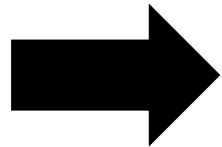
- La mémoire virtuelle utilisant la pagination est à une dimension
  - Les adresses virtuelles étant comprises entre 0 et une adresse maximale
- Il est parfois intéressant d'avoir plusieurs espaces d'adressage virtuel
  - Exemple d'un compilateur :
    - La table des symboles contenant le nom des variables
    - La table des constantes
    - L'arbre d'analyse syntaxique
    - La pile utilisée pour les appels de procédure du compilateur
- Une solution consiste à créer des espaces indépendants : **les segments**
  - On parle alors de **segmentation**

# La segmentation

## Introduction

Espace d'adressage  
virtuel

60K-64K	X
56K-60K	X
52K-56K	X
48K-52K	X
44K-48K	7
40K-44K	X
36K-40K	5
32K-36K	X
28K-32K	X
24K-28K	X
20K-24K	3
16K-20K	4
12K-16K	0
8K-12K	6
4K-8K	1
0K-4K	2



# La segmentation

## Définition

- Chaque segment est une suite d'adresses continues de 0 à une adresse maximale
  - Les segments sont en général de longueur différente
  - Les segments peuvent changer de taille en cours d'exécution
- Les segments sont des entités logiques que le programmeur doit manipuler
  - Un segment peut contenir une procédure, un tableau, etc.

# La segmentation

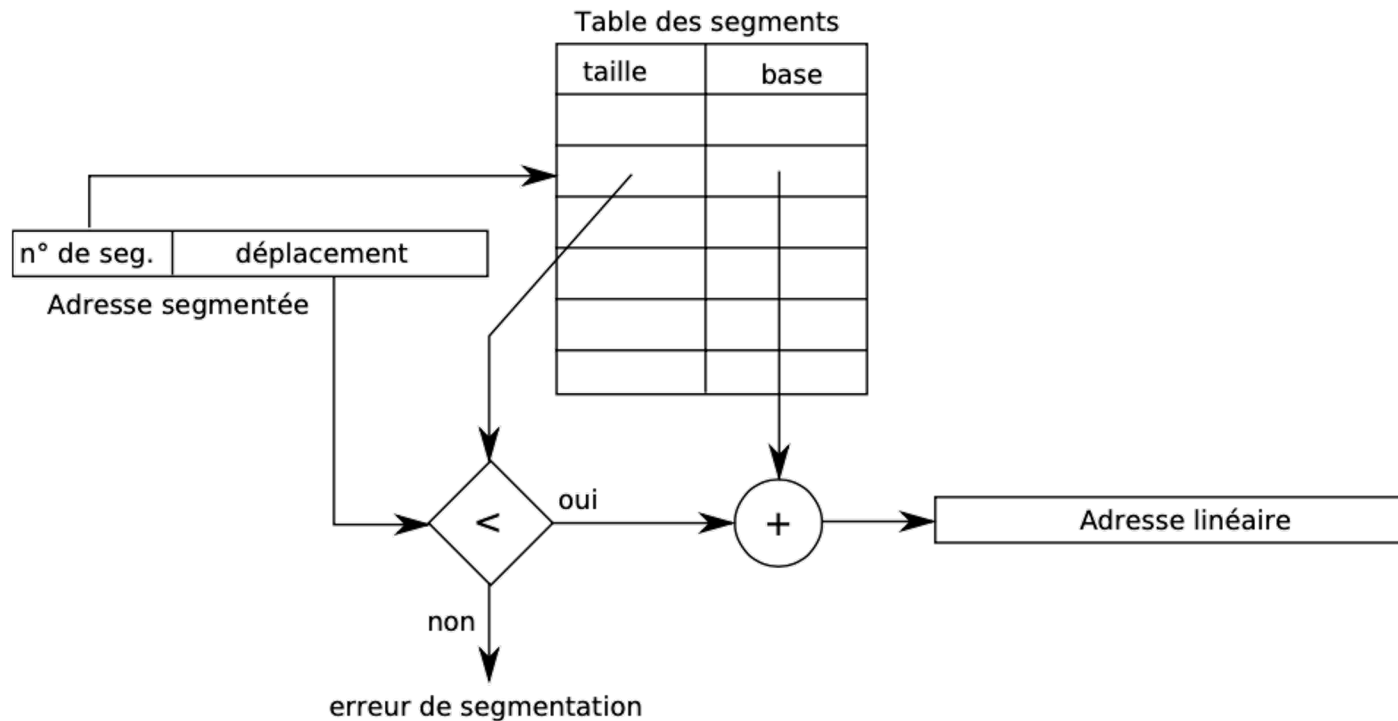
## Définition

- Un segment contenant une procédure peut être déclaré en exécution seule, ce qui interdit toute lecture ou écriture
- Une adresse est alors composée d'un couple
  - (numéro de segment, adresse dans le segment)
  - Lorsqu'un processus accède à une adresse d'un segment donné, le MMU vérifie que l'adresse est comprise dans le segment (inférieure à sa taille)
  - L'adresse est ajoutée à l'adresse de base du segment pour obtenir l'adresse finale

# La segmentation

## Conversion des adresses

- Mécanisme de conversion @segmentée -> @ linéaire



# La segmentation

## Avantages et inconvénient

- La segmentation offre les mêmes possibilités de partage et de sécurisation que la pagination
- Elle offre en plus une structuration logique de l'organisation
  - Partage de segment logique
    - Partage de variables entre plusieurs processus
  - Différentes propriétés de sécurité affectées aux segments de manière différentes
    - R-W-X
  - La taille des segments peut varier dynamiquement
- Inconvénient
  - La segmentation est sensible à la fragmentation externe

2
1
0

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?		
Combien y-a-t-il d'espaces linéaires ?		
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?		
Les procédures et les données peuvent-elles être séparées et protégées séparément ?		
Peut-on traiter facilement des tables variables ?		
Le partage de procédures entre utilisateurs est-il simplifié ?		

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?		
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?		
Les procédures et les données peuvent-elles être séparées et protégées séparément ?		
Peut-on traiter facilement des tables variables ?		
Le partage de procédures entre utilisateurs est-il simplifié ?		



# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?		
Les procédures et les données peuvent-elles être séparées et protégées séparément ?		
Peut-on traiter facilement des tables variables ?		
Le partage de procédures entre utilisateurs est-il simplifié ?		

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent-elles être séparées et protégées séparément ?		
Peut-on traiter facilement des tables variables ?		
Le partage de procédures entre utilisateurs est-il simplifié ?		

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent-elles être séparées et protégées séparément ?	Non	Oui
Peut-on traiter facilement des tables variables ?		
Le partage de procédures entre utilisateurs est-il simplifié ?		

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent-elles être séparées et protégées séparément ?	Non	Oui
Peut-on traiter facilement des tables variables ?	Oui	Oui
Le partage de procédures entre utilisateurs est-il simplifié ?		

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent-elles être séparées et protégées séparément ?	Non	Oui
Peut-on traiter facilement des tables variables ?	Oui	Oui
Le partage de procédures entre utilisateurs est-il simplifié ?	Non	Oui

# La segmentation

## Comparaison pagination / segmentation

Question	Pagination	Segmentation
Doit-on connaître la technique utilisée ?	Non	Oui
Combien y-a-t-il d'espaces linéaires ?	Un	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent-elles être séparées et protégées séparément ?	Non	Oui
Peut-on traiter facilement des tables variables ?	Oui	Oui
Le partage de procédures entre utilisateurs est-il simplifié ?	Non	Oui

- 🕒 **Pagination** : avoir un grand espace d'adressage sans avoir à augmenter la mémoire
- 🕒 **Segmentation** : permettre la séparation des programmes et des données dans l'espace d'adressage

# La segmentation

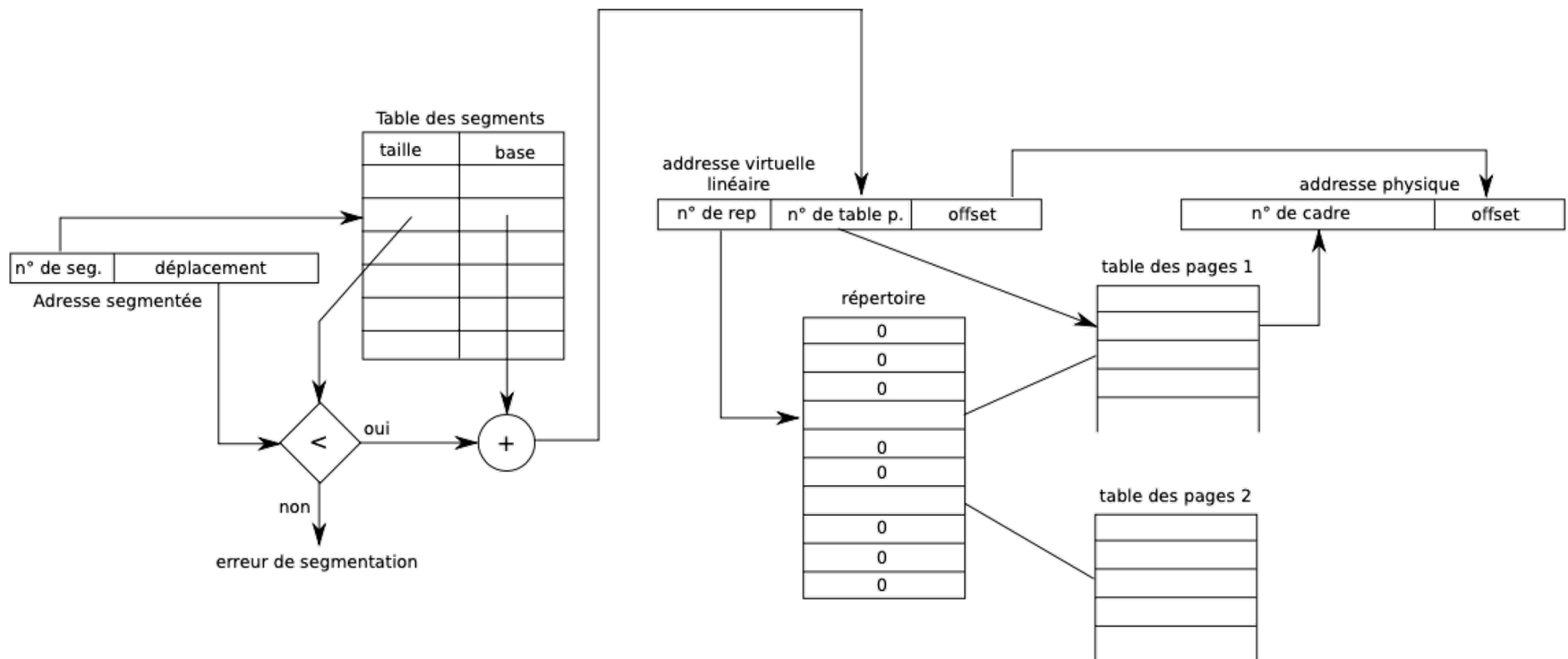
## La segmentation paginée

- La segmentation paginée
  - Combine les avantages des 2 approches
    - La segmentation offre une vue logique
    - La pagination élimine le problème de fragmentation externe
  - Les deux approches offrent des possibilités pour partager la mémoire
- Conversion des adresses :
  - Adresse logique : (n° de segment, déplacement)
  - -> adresse linéaire virtuelle (n° de répertoire, n° de page, déplacement)
  - -> adresse physique

# La segmentation

## La segmentation paginée

- Mécanisme de conversion :
  - @segmentée -> @ linéaire -> @ physique







Introduction

La gestion de la mémoire

La mémoire virtuelle

Les algorithmes de remplacement des pages

La segmentation

**Conclusion**

# Conclusion

## Conclusion

- La mémoire principale RAM
  - Permet au SE de stocker des informations
- Le gestionnaire de mémoire joue un rôle important
  - Espace d'adressage
- La stratégie va-et-vient
  - Les processus sont déplacés de la RAM au disque dur
- La mémoire virtuelle
  - Partitionnement de l'espace d'adressage virtuel
    - La pagination
- La segmentation
  - Permet d'organiser les données
  - Facilite le partage
- Les SE actuels utilisent une mémoire à segmentation paginée

## Et maintenant ?

- **On passe aux exercices !**
  - **Disponibles sur Moodle**
- N'oubliez pas à l'issue de la séance
  - Quizz
  - Feedback

# Systemes d'exploitation

## ENSISA 1A

### Chapitre 3

### Mémoire