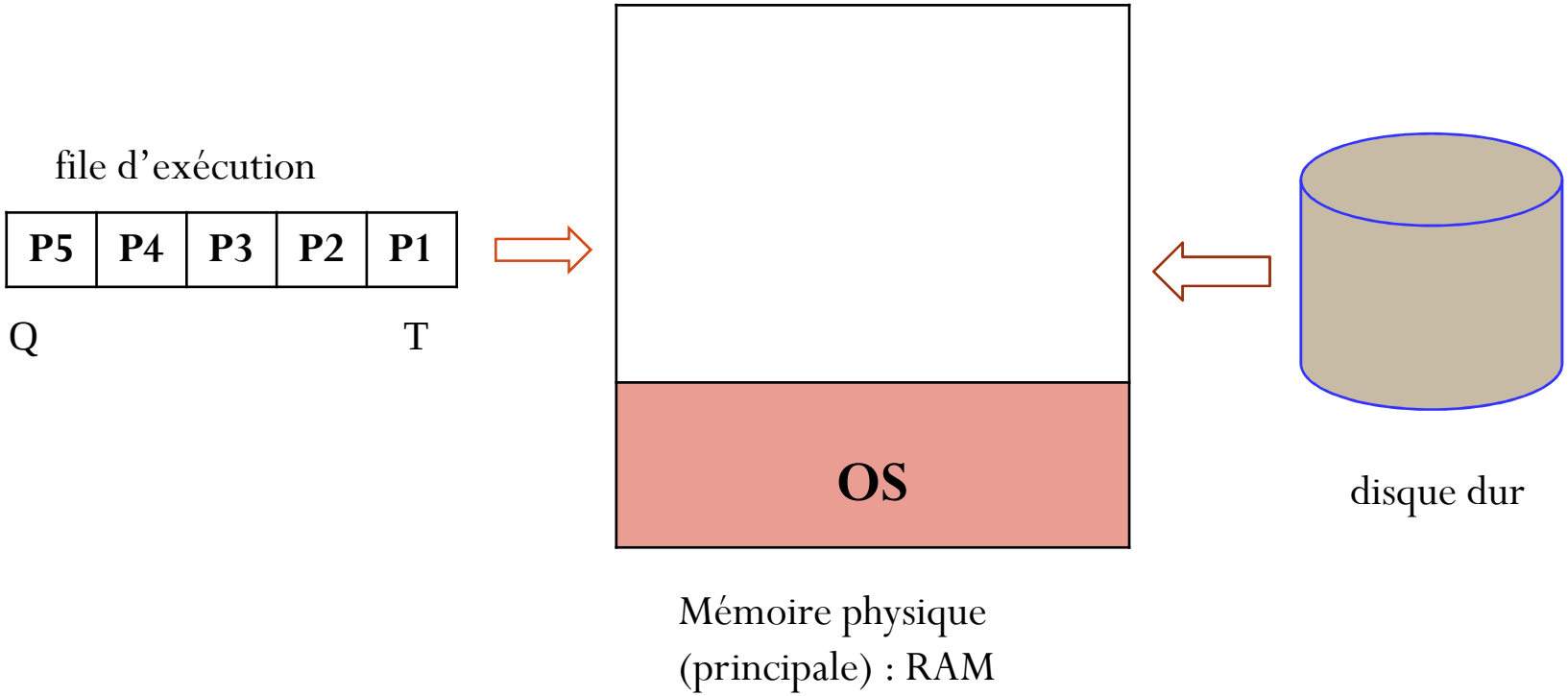
Systèmes d'exploitation

Chapitre 5: Gestion de la Mémoire

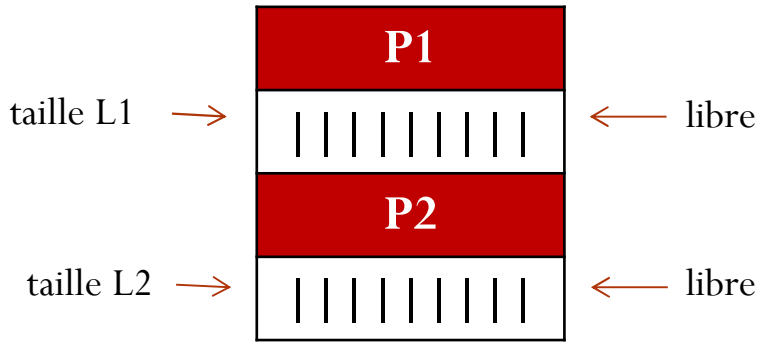
La Gestion de la Mémoire

1. Problématique:



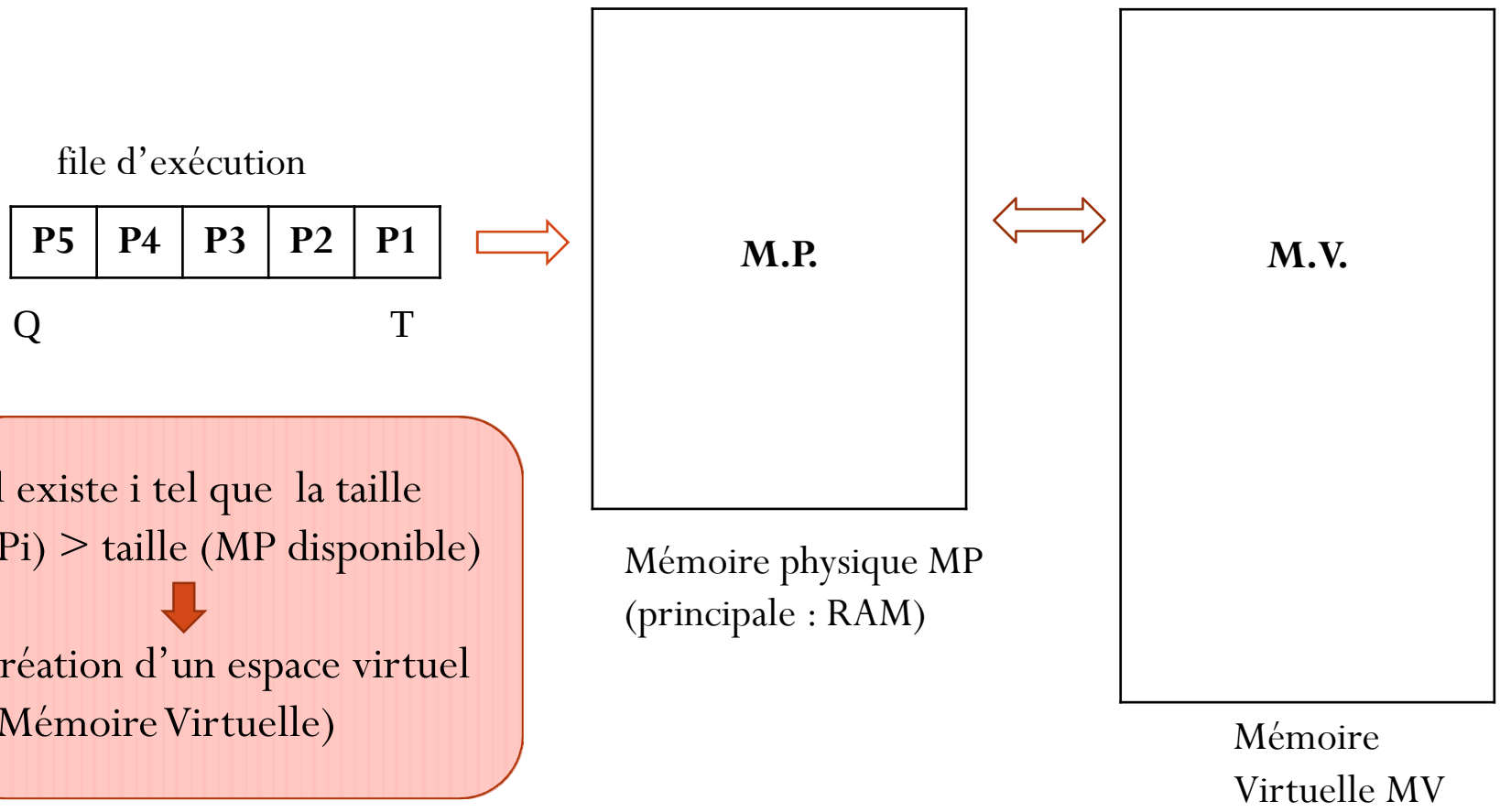
a) Trouver de la place mémoire pour charger les processus:

- Représentation (espaces libres)(structure de données)
- Algorithme de recherche
- Fragmentation



taille (P3) > L1
taille (P3) > L2
⇨ Taille (P3) ≤ L1+L2

b) Taille de la mémoire physique:



Il existe i tel que la taille (Pi) > taille (MP disponible)

⇓

création d'un espace virtuel (Mémoire Virtuelle)

Remarque: la taille (MV) >= 2* taille(MP)

Passage de la MV vers la MP: Etude de la Pagination

2. Représentation de la mémoire:

2.1. Représentation par Table de bits

- La mémoire est divisée en unités d'allocation dont la taille peut varier de quelques mots à plusieurs Kilos octets.
- Une unité représente une case mémoire.

Mémoire

A	A	A	| | |	| | |
| | |	| | |	B	B	B
| | |	| | |	| | |	C	C
C	C	C	| | |	| | |
D	D	D	D	| | |
| | |	| | |	E	E	E

unité= mot ou mots ou KO

Table de bits

1	1	1	0	0
0	0	1	1	1
0	0	0	1	1
1	1	1	0	0
1	1	1	1	0
0	0	1	1	1

$$\text{dimension de la table de bits} = \frac{taille\ (M.P.)}{taille\ (unié)}$$

A chaque unité, on fait correspondre un bit dans la table de bits

$$bit = \begin{cases} 0 & \text{si l'unité est libre} \\ 1 & \text{si l'unité est occupée} \end{cases}$$

A l'arrivé d'un processus:

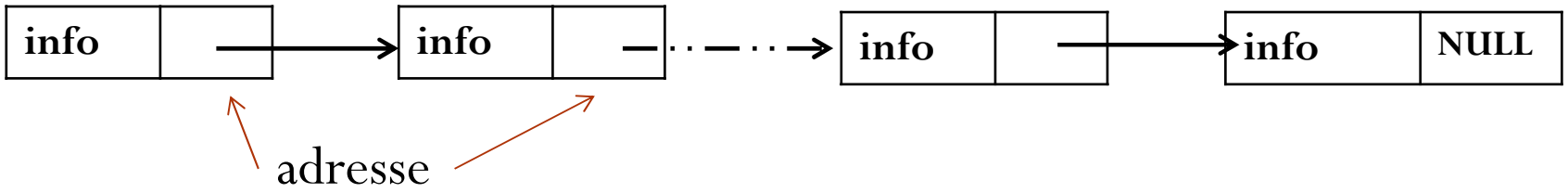
Le nombre d'unités nécessaire pour stocker le processus est:

$$\text{nb.unité(processus)} = \frac{taille(processus)}{taille(unité)} = k$$

Algorithme de recherche:

- Le gestionnaire de la mémoire doit alors parcourir la table de bits à la recherche de K unités consécutifs pour stocker le processus.
- Cet algorithme est très lent (inconvénient)

2.2. Représentation par la liste chaînée :



info = - atomique: int, float, char, double, …

- composée: structure.

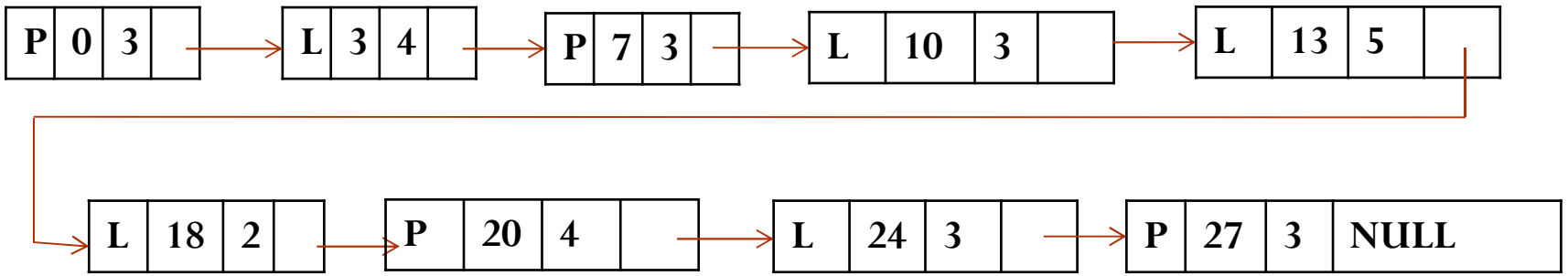
Au niveau mémoire, info est composé de:

1) L ou P: L pour libre et P pour occupée
2) Adresse de début
3) Taille : longueur

Exemple: La configuration suivante:

A	A	A	|||	|||
|||	|||	B	B	B
|||	|||	|||	C	C
C	C	C	|||	|||
D	D	D	D	|||
|||	|||	E	E	E

peut être représentée par:



Algorithme de recherche:

1) First Fit (première zone libre)
2) Best Fit (meilleur ajustement)
3) Worst Fit (plus grand résidu)

A l'arrivé d'un processus P, le nombre d'unités allouées est:

$$nb.unités = \frac{taille(processus)}{taille(unité)}$$

2.2.1: First Fit

Le gestionnaire de la mémoire parcourt la liste des segments à la recherche de la première zone mémoire libre qui peut contenir le processus (càd dont la taille est supérieure ou égale à la taille du processus).

C'est un algorithme très rapide, il provoque une fragmentation interne de la mémoire.

2.2.2: Best Fit

Le gestionnaire de la mémoire recherche dans toute la liste la plus petite zone libre qui peut contenir le processus.

C'est un algorithme long et provoque aussi une fragmentation de la mémoire(de très petites zones mémoires libres).

2.2.3:Worst Fit

Le gestionnaire de la mémoire parcourt la mémoire à la recherche de la plus grande zone libre qui peut contenir le processus.

C'est un algorithme long avec moins de fragmentation.

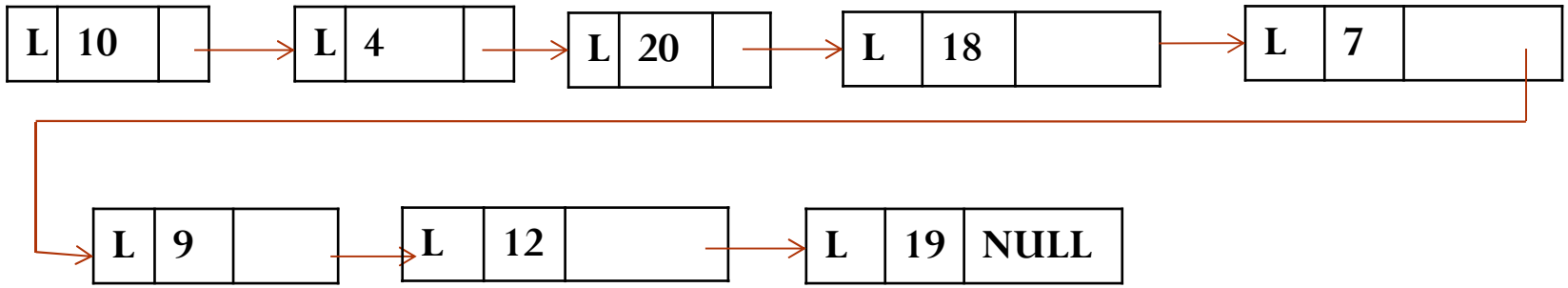
Application:

Si la liste des blocs libres contient les blocs de tailles 10k, 4k, 20k, 18k, 7k, 9k, 12k et 19k. Quelle sera la zone allouée pour les demandes de tailles 12k(A), 10k(B), 9k© et 8k(D) en utilisant les algorithmes:

- First Fit
- Best Fit
- Worst Fit

Solution:

Etat initial de la mémoire:



Etat final de la mémoire: ???

3. Pagination:

Problématique:

Dans le cas de la présence d'un processus dont la taille est supérieure à la taille de la mémoire physique disponible.

Solution:

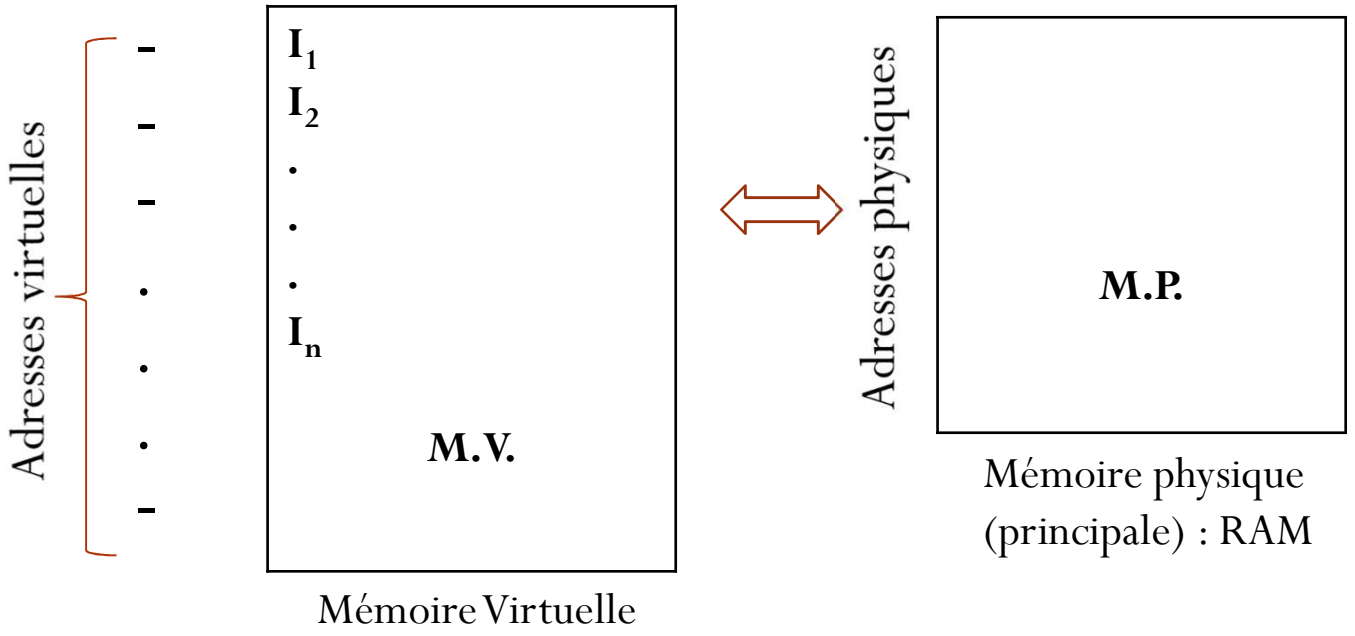
Mémoire virtuelle (pagination, segmentation, segmentation avec pagination)

Remarque:

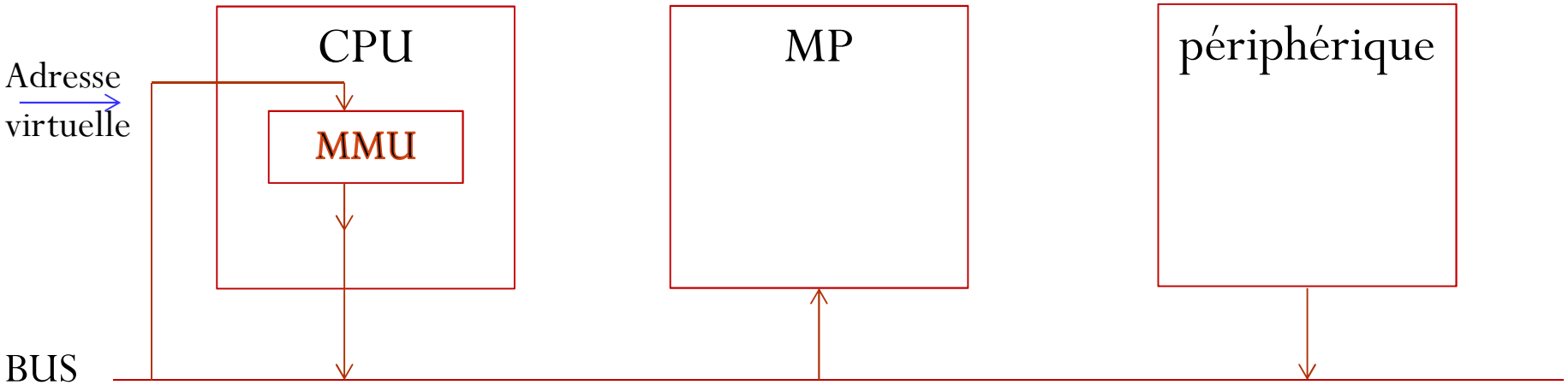
La taille (MV) ≥ 2*taille(MP)

3. 1. Principe de la pagination:

- La mémoire (virtuelle et physique) est divisée en zones de taille fixe, appelées pages.
- L'unité d'allocation est une page
- La taille d'une page varie entre 512 octets et quelques kilos octets



Transcodage d'une adresse virtuelle vers une adresse physique est réalisé par l'unité de gestion de mémoire (MMU: Memory Management Unit).



Emplacement et rôle de MMU

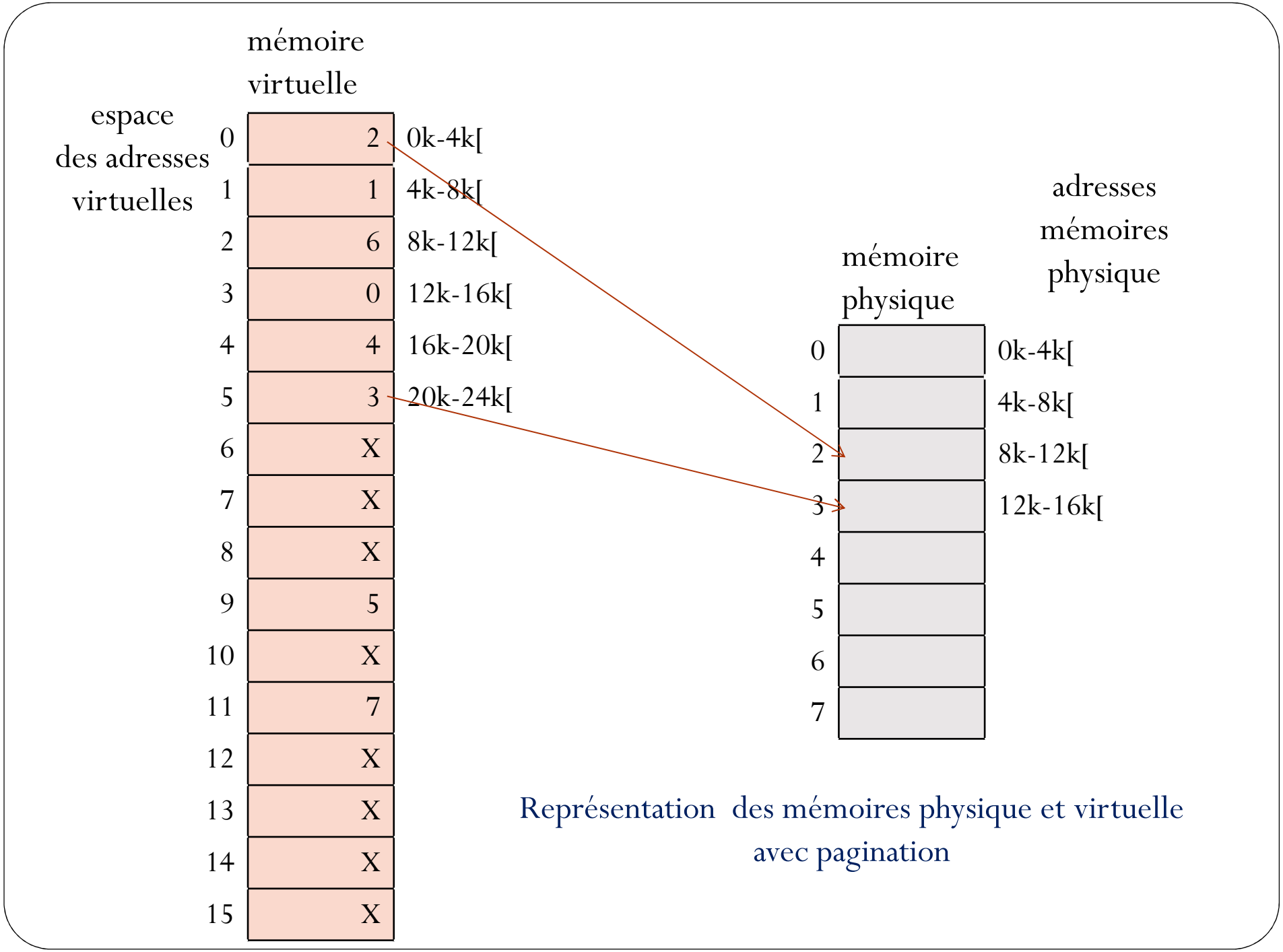
Etant donnée:

- une mémoire physique de taille 32 ko
- une mémoire virtuelle de taille 64 ko
- la taille d'une page est de 4 ko

Alors:

- le nombre de pages de la MP = $\frac{32\ ko}{4\ ko} = 8\ pages$

- le nombre de pages de la MV = $\frac{64\ ko}{4\ ko} = 16\ pages$



Représentation des mémoires physique et virtuelle avec pagination

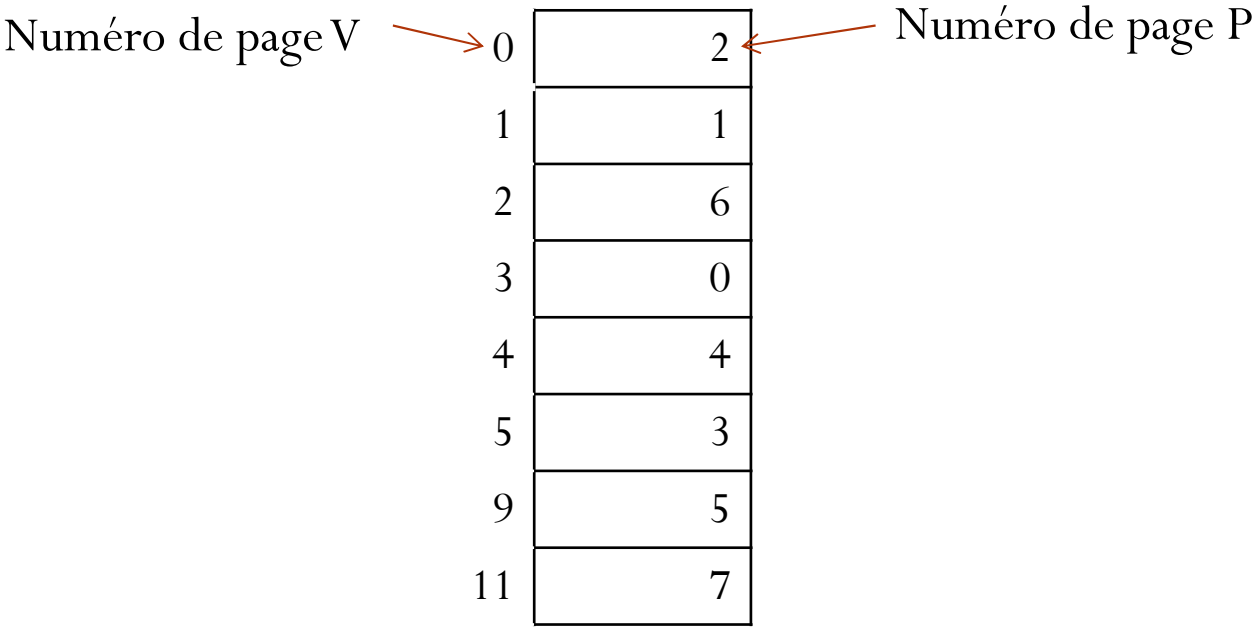


Table de pages décrivant la correspondance entre les adresses virtuelles et physiques

Adresse virtuelle:

N° de page virtuelle	Déplacement (offset)

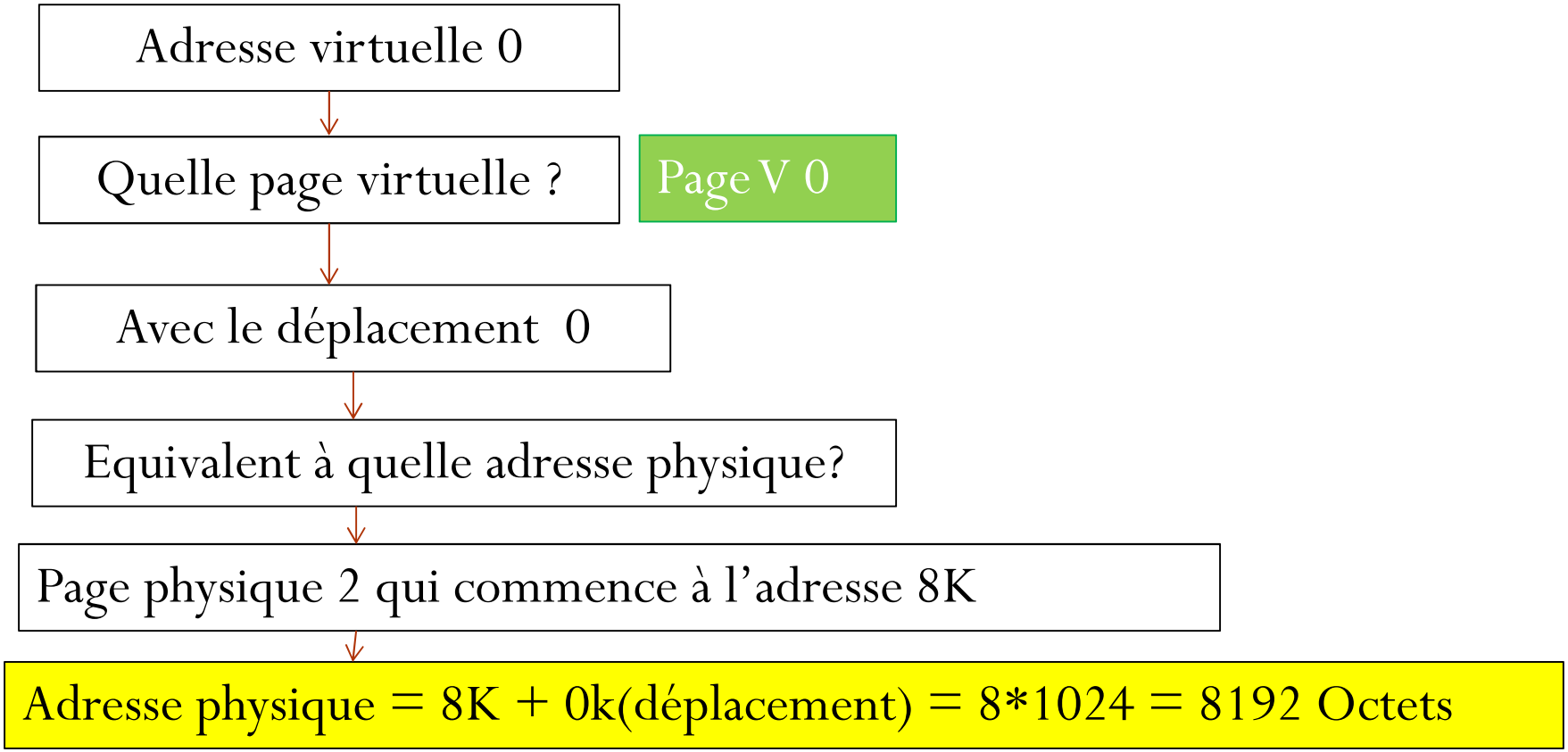
Adresse physique:

N° de page physique	Déplacement (offset)

- On note qu'une adresse virtuelle est composée du numéro de page virtuelle plus le déplacement dans celle-ci.
- Pour chaque adresse virtuelle, le MMU fait correspondre une adresse physique composée du numéro de page physique et d'un déplacement qui est égale au déplacement dans la page virtuelle.
- La problématique est de trouver l'équivalent d'une page virtuelle vers une page physique.

Exemple:

Calculer l'adresse physique qui correspond à l'adresse virtuelle 0

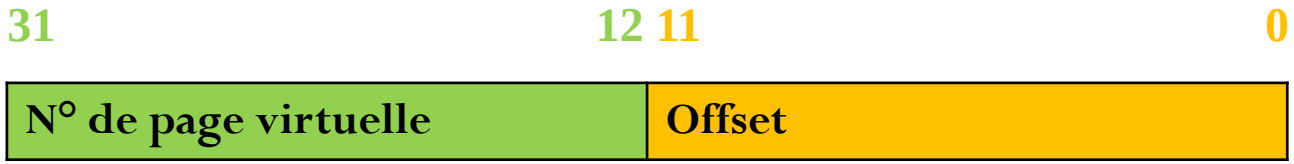


Application:

Calculer l'adresse physique qui correspond à l'adresse virtuelle 12500 Octets? et de 20660 Octets?

Ecriture des Adresses

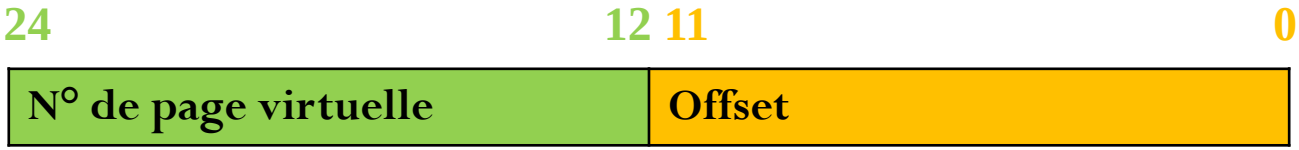
Une adresse virtuelle de 32 bits pourrait être découpée en un champ de 20 bits et un champ de 12 bits :



✓Ceci représente 1 M (2^{20}) pages de 4 Ko (2^{12} o).

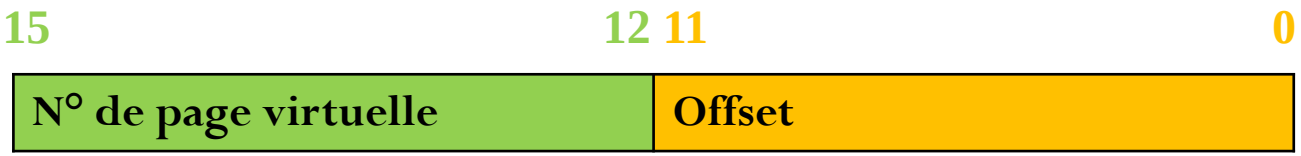
✓Ce qui donne un total de 4 Go (2^{32} o).

Supposons que notre ordinateur a une mémoire physique de 32 Mo (2^{25} o) . Il aura donc 8 K (2^{13}) pages de 4 Ko. L'adresse physique aura donc la forme suivante:



Exemple 1:

- si la taille (Mémoire Virtuelle) = 64 ko = $2^{6} \times 2^{10}$ o $= 2^{16}$ o. d´où l´utilisation de 2^{16} adresses virtuelles.
- chacune peut être codée sur 16 bits.
- si la taille (Page Virtuelle) = 4 ko = $2^{2x}\ 2^{10}$ o = 2^{12} o, ce qui donne 12 déplacement.
- chacun d´entre eux peut être codé sur 12 bits de la façon suivante:



Notons que:

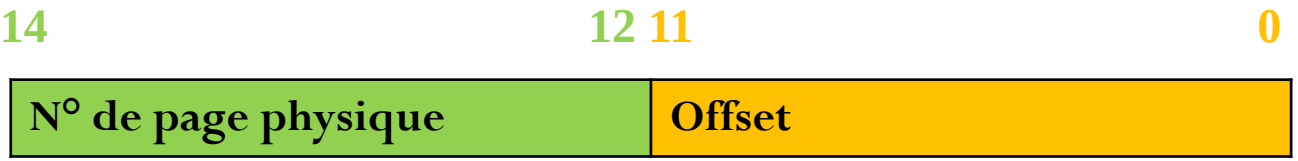
- Nombre de page virtuelle = taille (M V) / taille (Page)
  = 64 ko / 4 ko = 16 pages = 2^4 pages virtuelle.

Chaque N° de page virtuelle peut être codé sur 4 bits.

Exemple 2:

- si la taille (Mémoire Physique) = 32 ko = 2^5 x 2^{10} o = 2^{15} o, d´où l´utilisation de 2^{15} adresses physique.
- chacune peut être codée sur 15 bits de la façon suivante:

Si la taille d´une page physique est égale toujours à 4 ko, on aura:



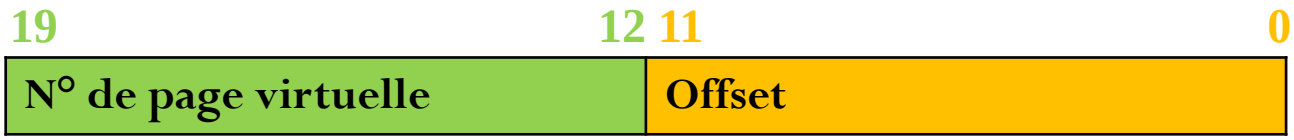
Notons que:

- Nombre de page physique = taille (M P) / taille (Page)
  = 32 ko / 4 ko = 8 pages = 2^3 pages physique.

Chaque N° de page physique peut être codé sur 3 bits.

Application :

Une adresse virtuelle paginée nécessite 20 bits organisés comme suit:



Question: Quelle est la taille de la mémoire virtuelle exprimée en Octet et en nombre de pages?

Réponse:

- Taille (MV) = 2^{20} o = 2^{10} x 2^{10} o = 1 Mo
- NB(PV) = taille (MV) /taille(PV) = 2^{20} o / 2^{12} o = 2^{8} pages.
- Taille (M en page) = 256 pages.

Application :

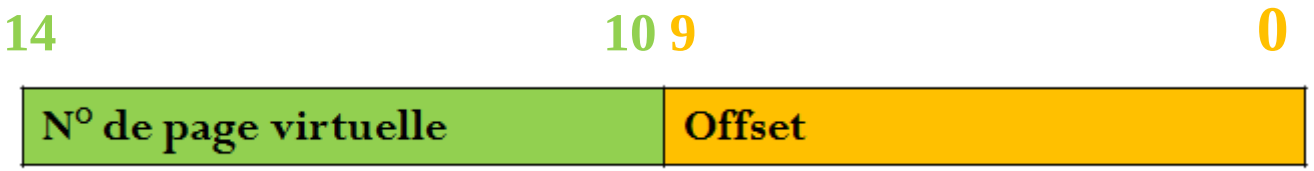
Soit un espace physique de 8 pages de 1024 octets, chacune en correspondance avec une mémoire logique (virtuelle) de 32 pages:

Question:

- combien de bits existent-ils dans une adresse physique,
- combien de bits existent-ils dans une adresse logique,

Réponse:

- Taille (MV) = 32x1ko = 2^5 x 2^{10} o = 2^{15} o



- Taille (MP) = 8x1ko = 2^3 x 2^{10} o = 2^{13} o

Table de page

N° de pages Virtu_elle	N° de page Physique	Bits complémentaires
0	2	
1	1	
2	6	
3	0	
4	4	
5	x	
6	x	
7		
.		
.	x	
15	x	

Bits complémentaires:

- 1 bit de présence,
- protection (r: lecture, w: écriture, x: exécution)

✓La table de pages permet de trouver le correspondant d´une page virtuelle vers une page physique.

✓Pour chaque index (N° de page Virtuelle) on fait correspondre le N° de page Physique + des bits complémentaires composés d´un bit de présence (1: si la page virtuelle est chargée en mémoire physique, 0 : sinon) + des bits de protection qui définissent les droits de lecture, écriture et exécution.

Remarques:

✓Généralement, le nombre de pages Virtuelles est égale au moins 2 fois le nombre de pages physiques,

✓ Donc quelques pages Virtuelles ne seront pas mappées (n´auront pas de correspondance en mémoire physique),

✓ Un défaut de page (déroutement) se produit lorsque le SE demande une page virtuelle non mappée.

✓Dans ce cas là, le SE:

- Repère une page physique peu utilisée,
- Recopie son contenu sur le disque,
- Place la page demandée dans la case libérée ,
- Modifie la table de pages,
- Exécution de l´instruction,

Taille de la table de pages:

- Taille (TP) = Nombre de pages Virtuelle * (Nombre de bits pour coder un N° de page physique + bits complémentaires).
- Dans notre exemple :

T = 16 * (3 + bits complémentaires)

Exercice1:

Soit un espage d´adressage virtuelle de 4 Go associé à un espace physique de 32 Mo. La taille d´une page Virtuelle est de 1 Ko.

1) Ecrire les adresses virtuelles et physiques.
2) Calculer la taille de la table de page si elle comporte seulement 1 bit complémentaires

Exercice2:

- Une adresse virtuelle s´écrit comme suit:

Et la taille de la mémoire physique est égale à 16 Mo.

Questions:

1) Donner le nombre de pages Virtuelles et Physiques
2) Donner l´écriture adresse physique
3) Calculer la taille de la table de page (si on suppose qu´on dispose de 4 bits complémentaires).

Solution:

c-à-d: 24 bits pour coder une adresse physique

1) Nombre de pages Virtuelles = $2^{33} / 2^{16} = 2^{17}$ pages
   Nombre de pages Physiques = $2^{24} / 2^{16} = 2^{8}$ pages

1) Donner l´écriture adresse physique

1) La taille de la table de page = $2^{17} * (8 + 4) = 2^{17} * 12$ bits