La gestion de la mémoire

Objectifs de ce module

En suivant ce module vous allez:

- **Présenter les mécanismes :** Segmentation et pagination.

La mémoire : coquille des programmes.

Organisation de la mémoire physique : Les mécanismes nécessaires.

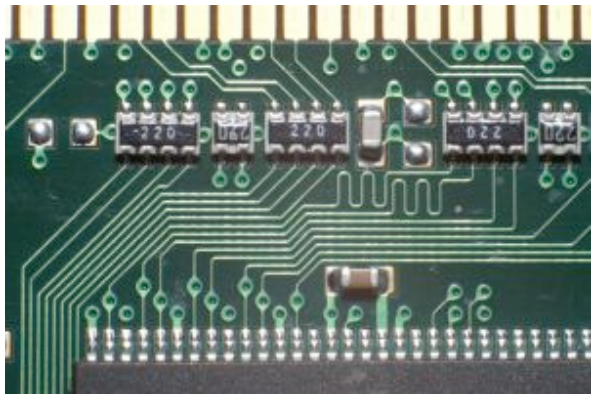
La mémoire

Stocker les données et le travail intermédiaire

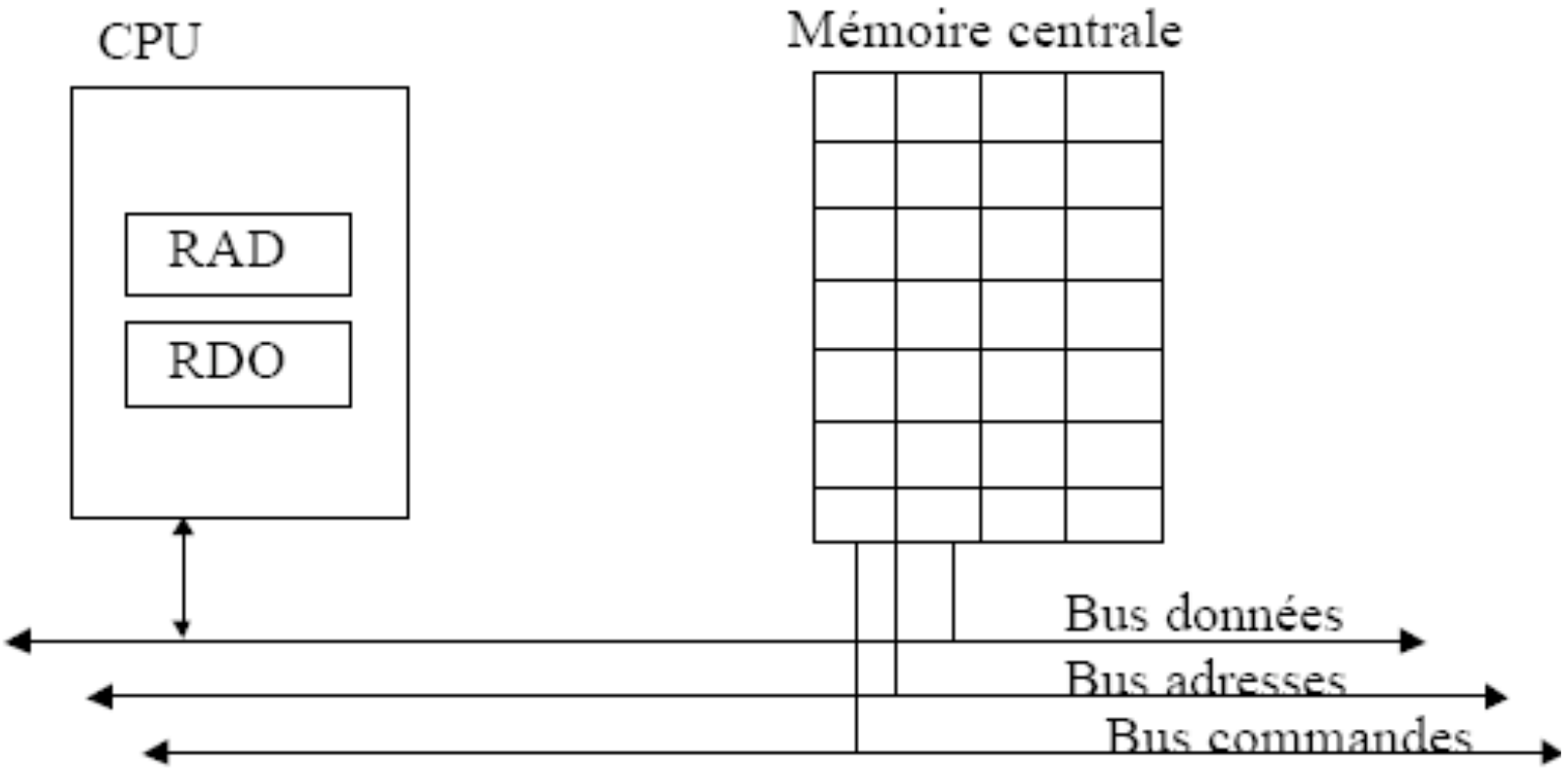
Plan de la partie

Voici les chapitres que nous allons aborder:

- La fonction de mémorisation.
- Le chargement dynamique en mémoire
- Le découpage de la mémoire.
- Le partage de la mémoire.
- Organisation de la mémoire d'un processus



La fonction de mémorisation



ensemble linéaire de mots d'adresses contiguës

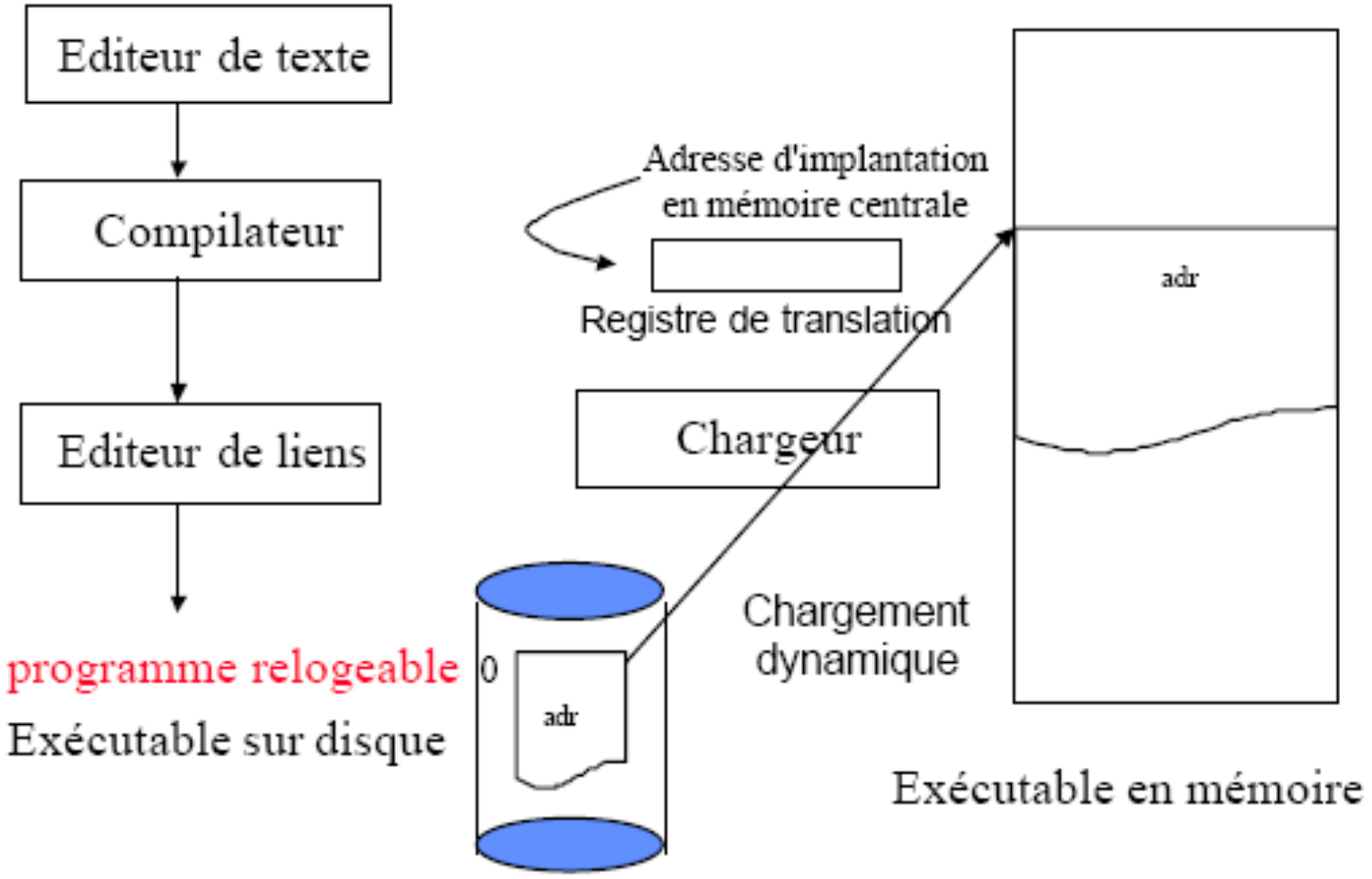
La fonction de mémorisation

Trois problèmes à résoudre vis-à-vis de la mémoire :

- Définir un espace d'adressage indépendant pour chaque processus
- Protéger les espaces d'adressages des processus entre eux
- Allouer de la mémoire physique à chaque espace d'adressage

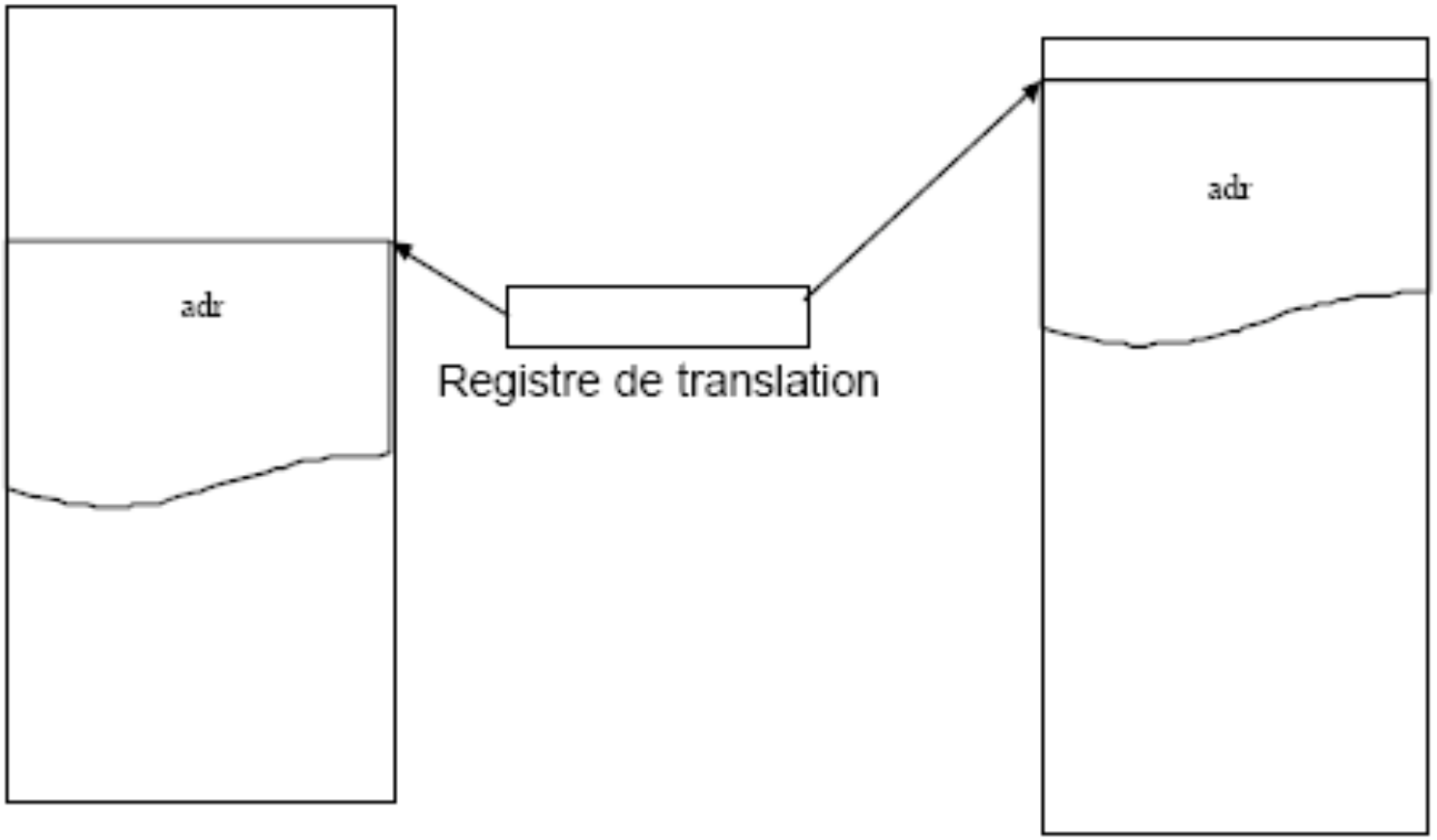
Le chargement dynamique

les adresses du programme sont
translatées de la valeur de l'adresse d'implantation en MC à l'exécution



Le chargement dynamique

Déplacer un programme : modifier la valeur du registre de translation

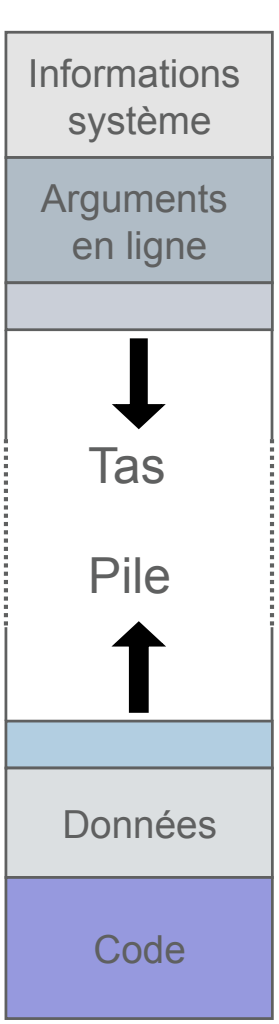


Le découpage de la mémoire

La mémoire centrale peut-être découpée de trois façons :

- la **segmentation** : les programmes sont découpés en parcelles ayant des longueurs variables appelées «segments».
- la **pagination** : elle consiste à diviser la mémoire en blocs, et les programmes en pages de longueur fixe.
- une combinaison de **segmentation** et de **pagination** : certaines parties de la mémoires sont segmentées, les autres sont paginées.

Organisation de la mémoire d'un processus



L'espace mémoire utilisé par un processus est divisé en plusieurs parties représentées sur la figure ci dessous.

On trouve en particulier le segment de code (souvent appelé segment de text), le segment de données, la pile (stack en anglais) et le tas (heap en anglais).

Chaque processus croit être le seul à occuper la mémoire et pense que celle-ci a une taille de 4 Go.

Organisation de la mémoire d’un processus

Le segment de code

Le segment de code est obtenu en copiant directement en mémoire le segment de code du fichier exécutable.

Au cours de l'exécution du programme, la prochaine instruction à exécuter est repérée par un pointeur d'instruction.

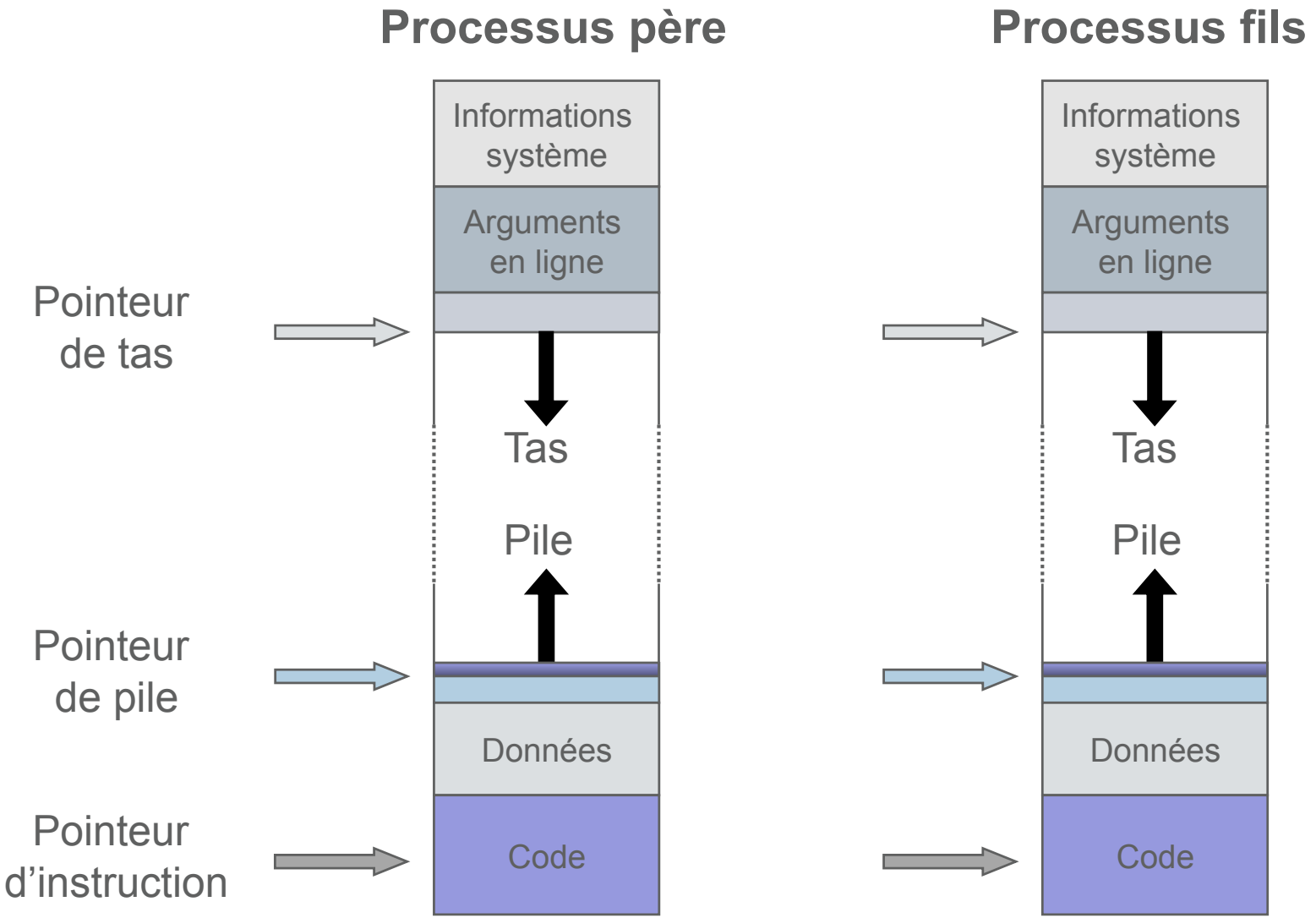
Organisation de la mémoire d’un processus

Le segment de données

Au dessus du segment de code se trouve le segment de données.

Ce segment est traditionnellement composé d'un segment de données initialisées (*data*), qui est directement copié à partir de l'exécutable, et d'un segment de données non initialisées (*bss* pour *block storage segment*) qui est créé dynamiquement.

Organisation de la mémoire d'un processus



Organisation de la mémoire d'un processus

Les données initialisées correspondent à toutes les variables globales et statiques initialisées des programmes C. Les données non initialisées correspondent aux variables globales et statiques non initialisées.

Le segment de données est agrandi ou réduit au cours de l'exécution pour permettre d'y placer des données. Néanmoins, il est habituel de considérer que ce segment est fixe et correspond à celui obtenu avant l'exécution de la première instruction : le segment *bss* se résume alors aux variables locales de la fonction main().

La pile et le tas sont des zones variables.

Organisation de la mémoire d'un processus

Les autres informations

D'autres informations sont placées dans l'espace mémoire du processus, comme les paramètres passés en ligne lors de l'exécution du programme.

Suivant les systèmes d'exploitation, le système peut aussi stocker différents renseignements, comme par exemple le mode dans lequel doit s'exécuter le processus. Ces renseignements se trouvent dans une structure qui se nomme souvent *Process Control Bloc* (PCB).

Organisation de la mémoire d’un processus

Conclusion

L'espace mémoire d'un processus est donc divisé en deux morceaux variables situés chacun à une extrémité de l'espace. Ces morceaux sont eux-mêmes divisés en plusieurs segments, certains de taille fixe situés aux extrémités, d'autres de taille variable.

Suivant les systèmes d'exploitation, le tas et la pile peuvent se trouver l'un en haut et l'autre en bas, ou l'inverse.

Organisation de la mémoire physique

Mécanismes

Plan de la partie

Voici les chapitres que nous allons aborder:

- Pagination.
- Pagination et Segmentation
- Segmentation.
- Segmentation et Pagination.
- Mémoire virtuelle.

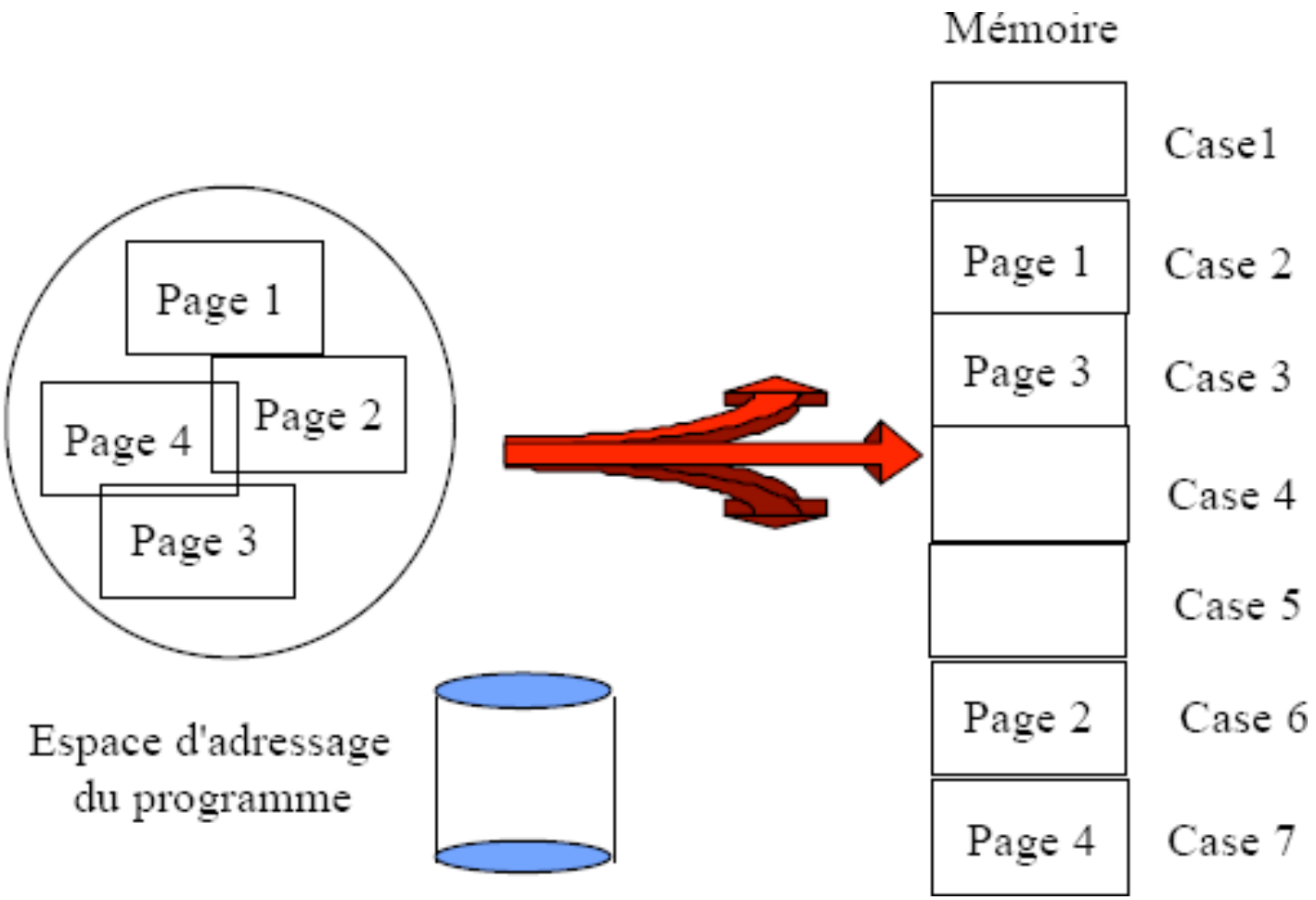
Pagination

La pagination constitue une approche permettant de réduire la fragmentation de la mémoire (fragmentation externe).

L'espace des adresses logiques d'un processus est 'paginé' :

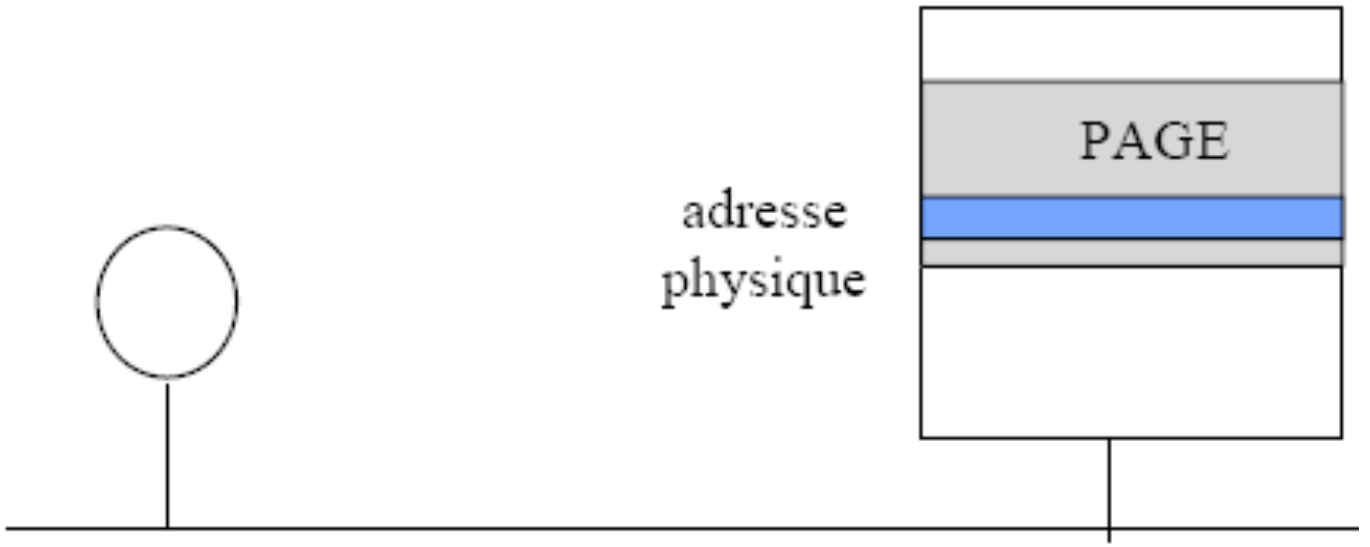
- Les pages sont indépendantes en terme d'adressage à l'intérieur de la mémoire et ne sont pas nécessairement contiguës
- Le SE alloue la mémoire physique chaque fois que la nécessité de loger une page se présente.

Pagination


Mémoire
Case1
Page 1
Case 2
Page 3
Case 3
Case 4
Case 5
Page 2
Case 6
Page 4
Case 7
Page 1
Page 2
Page 4
Page 3
Espace d'adressage
du programme

Pagination

De 512 octets
à
8192 octets

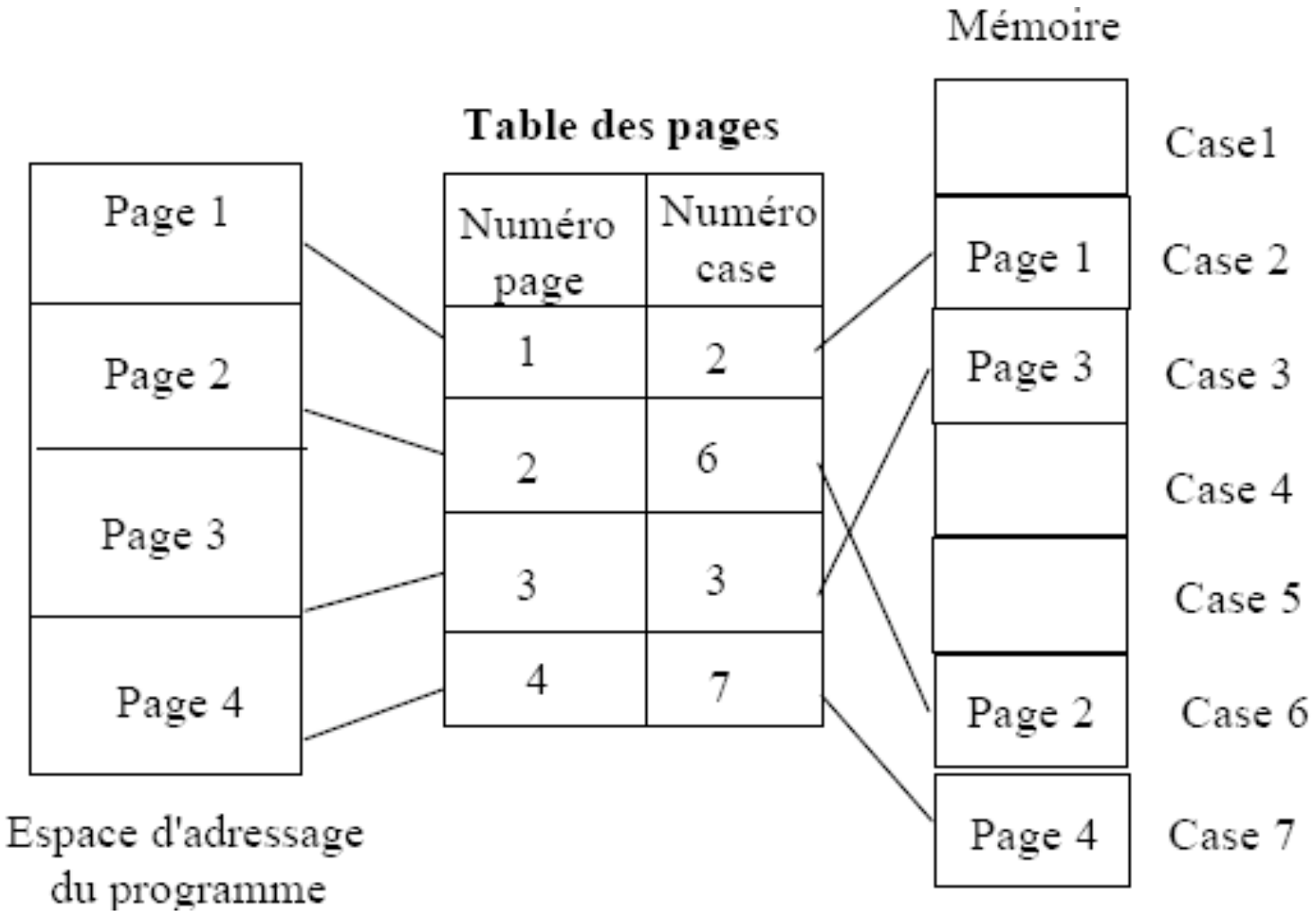


<n°page; deplacement>

Il faut convertir l'adresse paginée en son équivalent adresse physique
Adresse physique =
adresse implantation case contenant la page + deplacement

Translation garantie par la ***Table des pages***

Pagination


Mémoire
Table des pages
Numéro page
Numéro case
1
2
2
6
3
3
4
7
Page 1
Page 2
Page 3
Page 4
Espace d'adressage du programme
Case1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Page 1
Page 3
Page 2
Page 4

Pagination

Implémentation de la table des pages : 2 approches

Un ensemble de registres matériels

– commuter de processus = charger tous les registres avec les adresses des pages du processus

– la table des pages ne peut pas être très grande (256 entrées)

En mémoire centrale, repérée par un registre PTBR (*page table base register*)

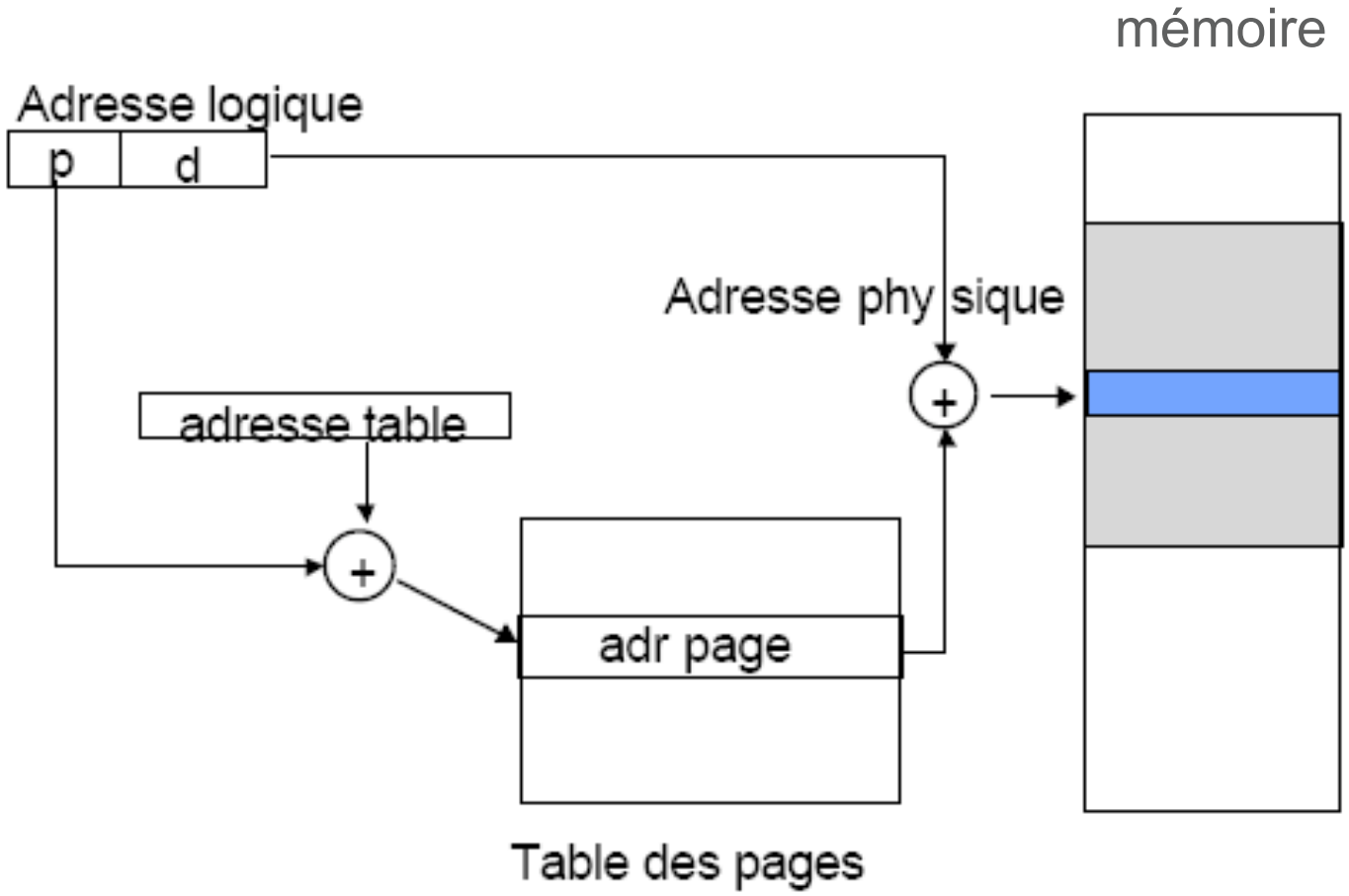
– commuter de processus = charger le registre PTBR avec l'adresse de la table des pages du processus

– pas de limite de taille à la table des pages

– problème : le temps d'accès à un emplacement mémoire d'un programme

Pagination

temps d'accès à un emplacement mémoire d'un programme :
deux accès mémoire

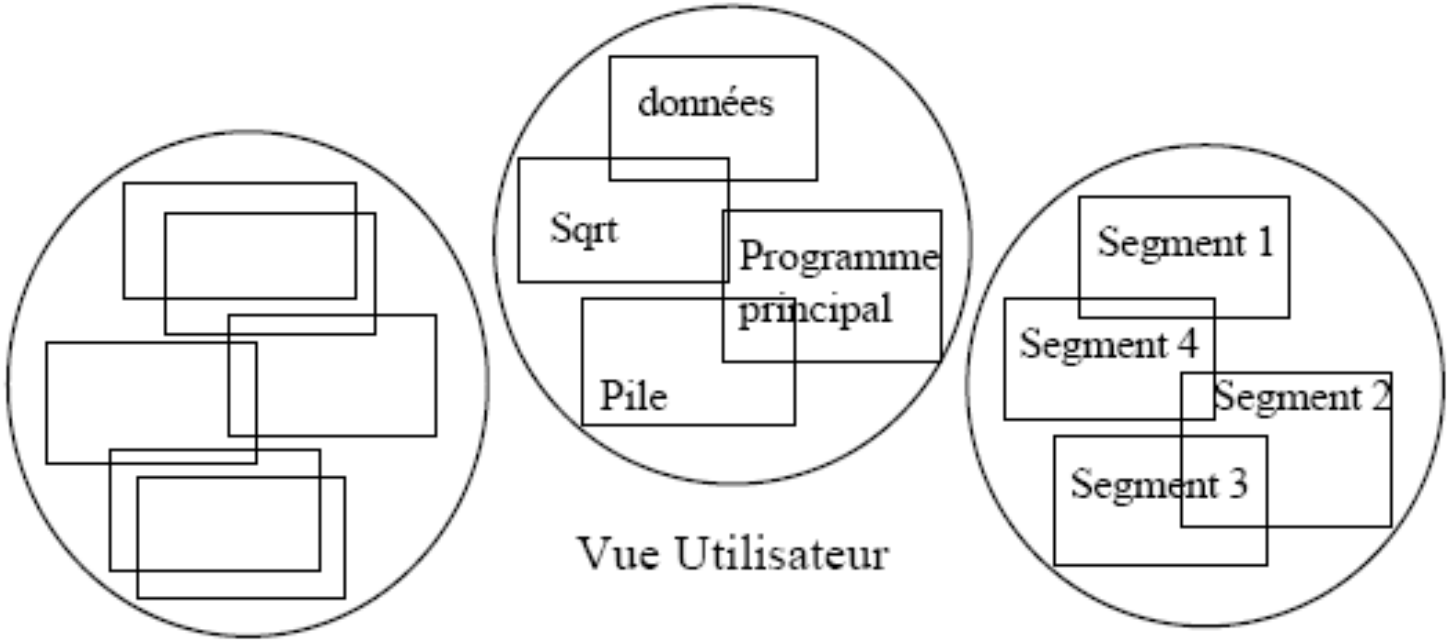


Pagination

Avantages et inconvénients:

- Meilleure utilisation de la mémoire physique (programmes implantés par fragments, dans des pages *non consécutives*).
- Possibilité de ne charger des pages que lorsqu'elles sont référencées (chargement à la demande).
- Indépendance de l'espace virtuel et de la mémoire physique (mémoire virtuelle généralement plus grande).
- Par contre, fragmentation interne (toutes les pages ne sont pas remplies).

Pagination et Segmentation



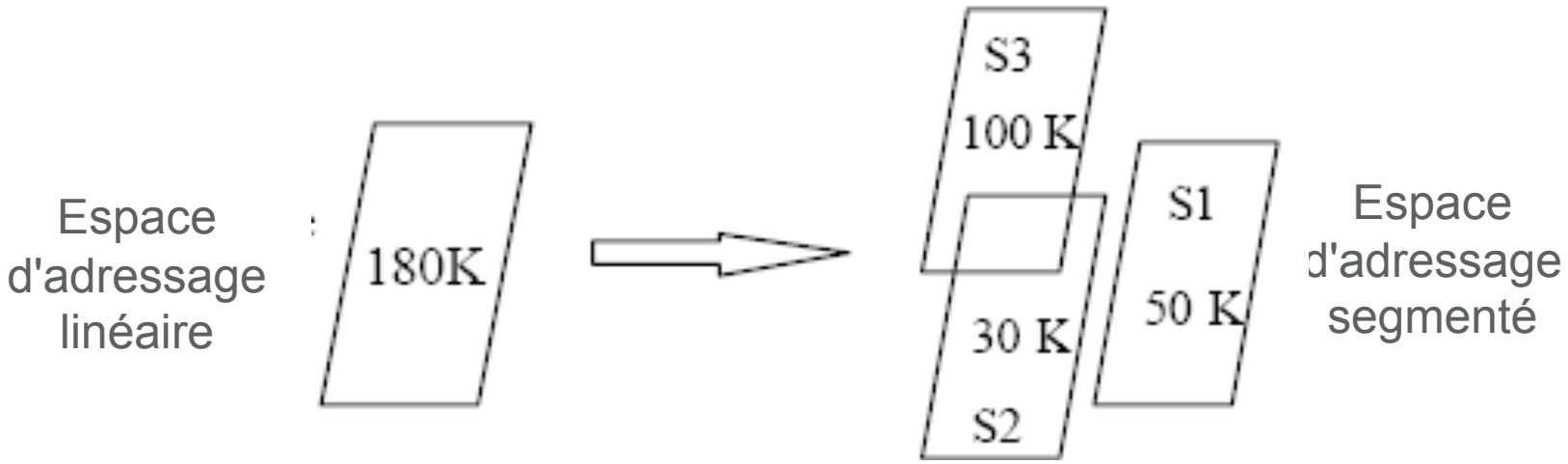
Espace d'adressage
du programme
Ensemble de pages

Espace d'adressage
du programme
Ensemble de segments

Segmentation

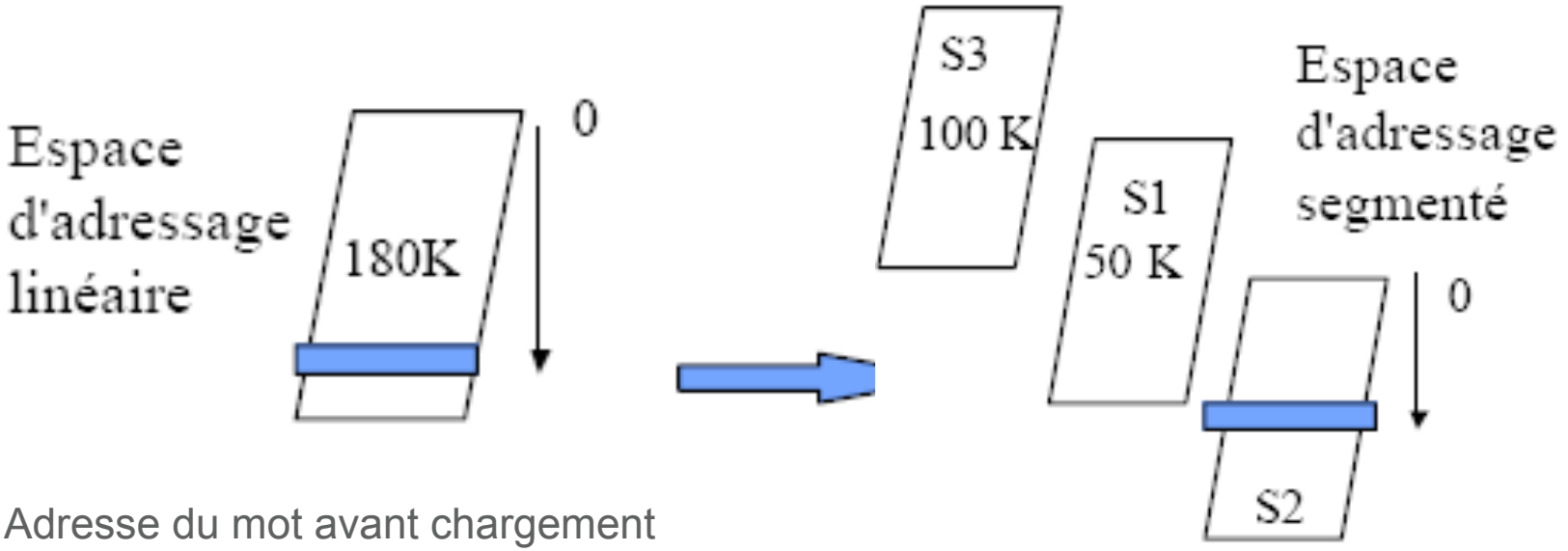
Le programme est divisé en segments par le compilateur, un segment étant un espace d'adressage linéaire formé d'adresses contigues.

Un segment correspond à une partie logique du programme : segment de code, segment de données, segment de pile



Organisation de la mémoire physique

Segmentation



Adresse du mot avant chargement
Déplacement depuis 0

Adresse du mot après chargement :

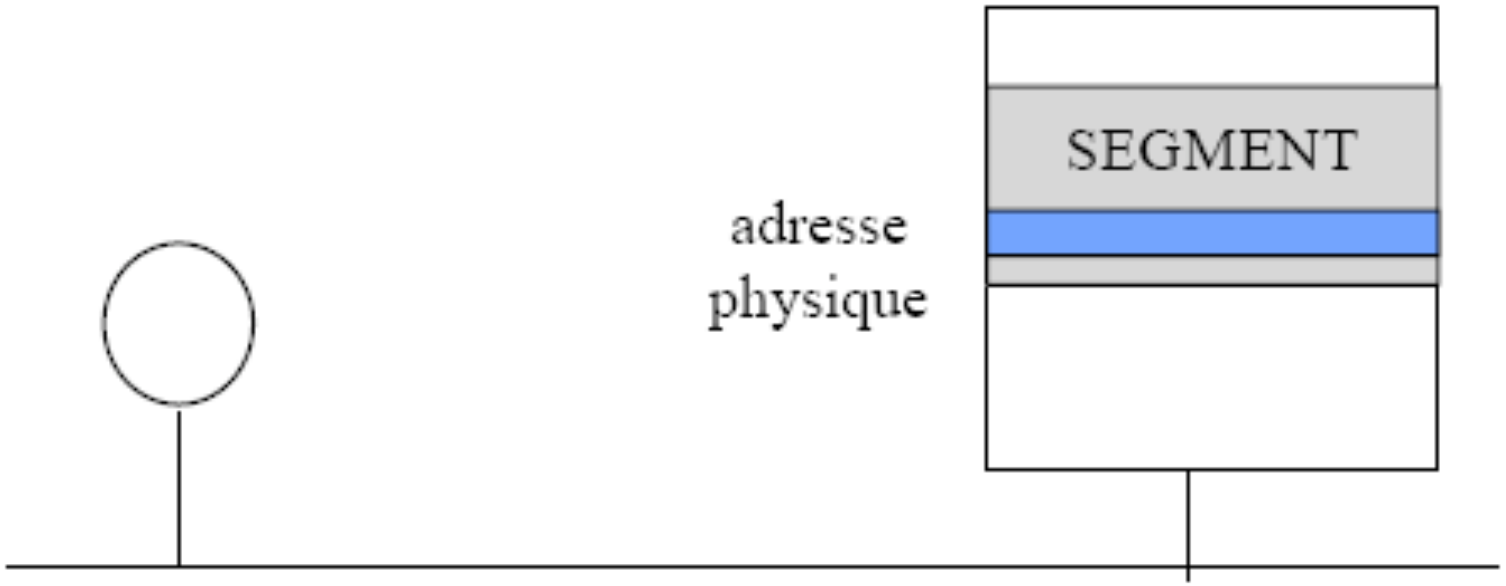
Adresse d'implantation + Déplacement

Adresse du mot avant chargement
N° segment, Déplacement depuis 0

Adresse du mot après chargement :

Adresse d'implantation du segment
+ Déplacement

Segmentation



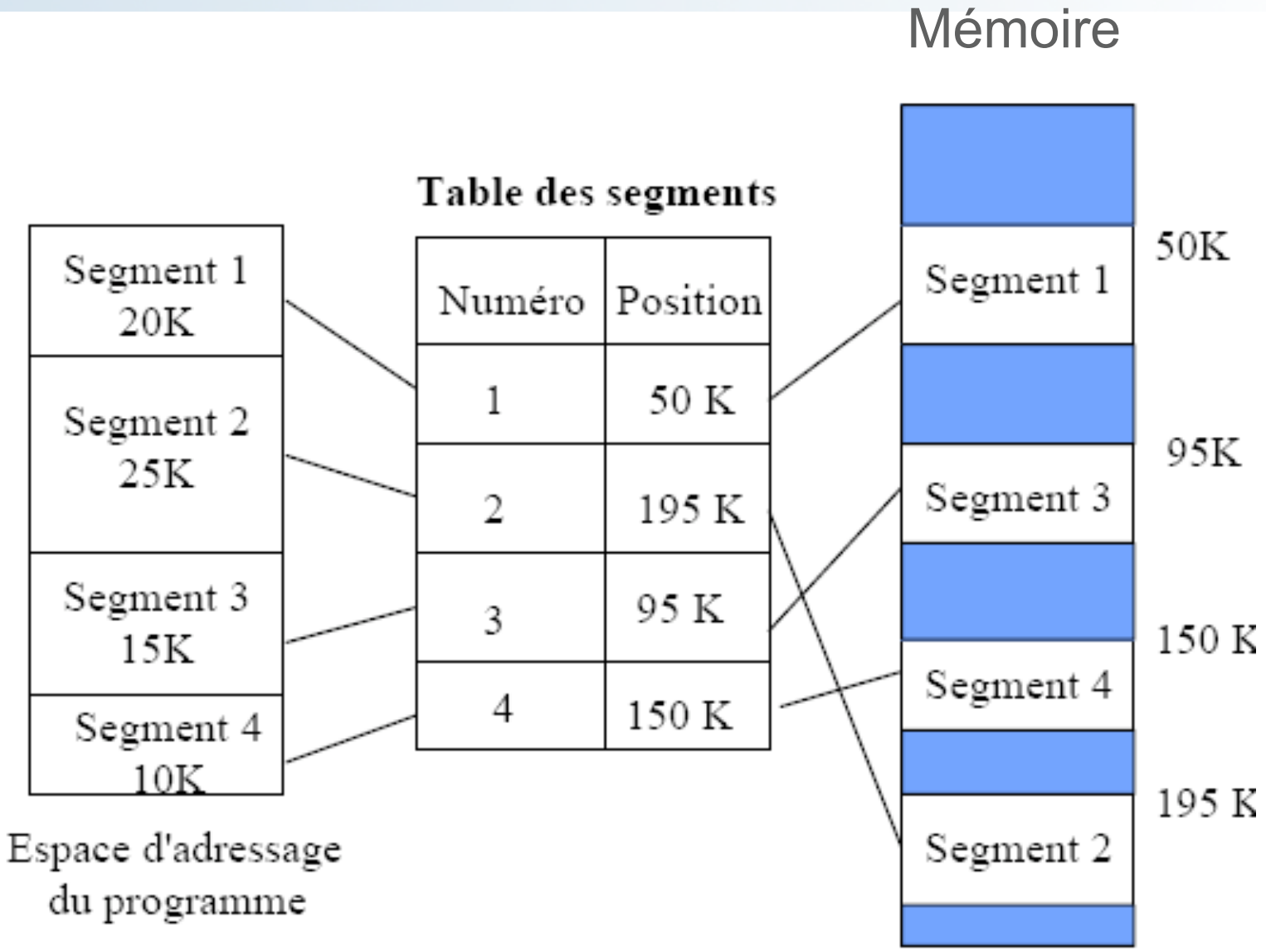
<n°segment; deplacement>

Il faut convertir l'adresse segmentée en son équivalent adresse physique

Adresse physique =
adresse implantation segment + deplacement

Translation garantie par la **Table des segments**

Segmentation



Segmentation

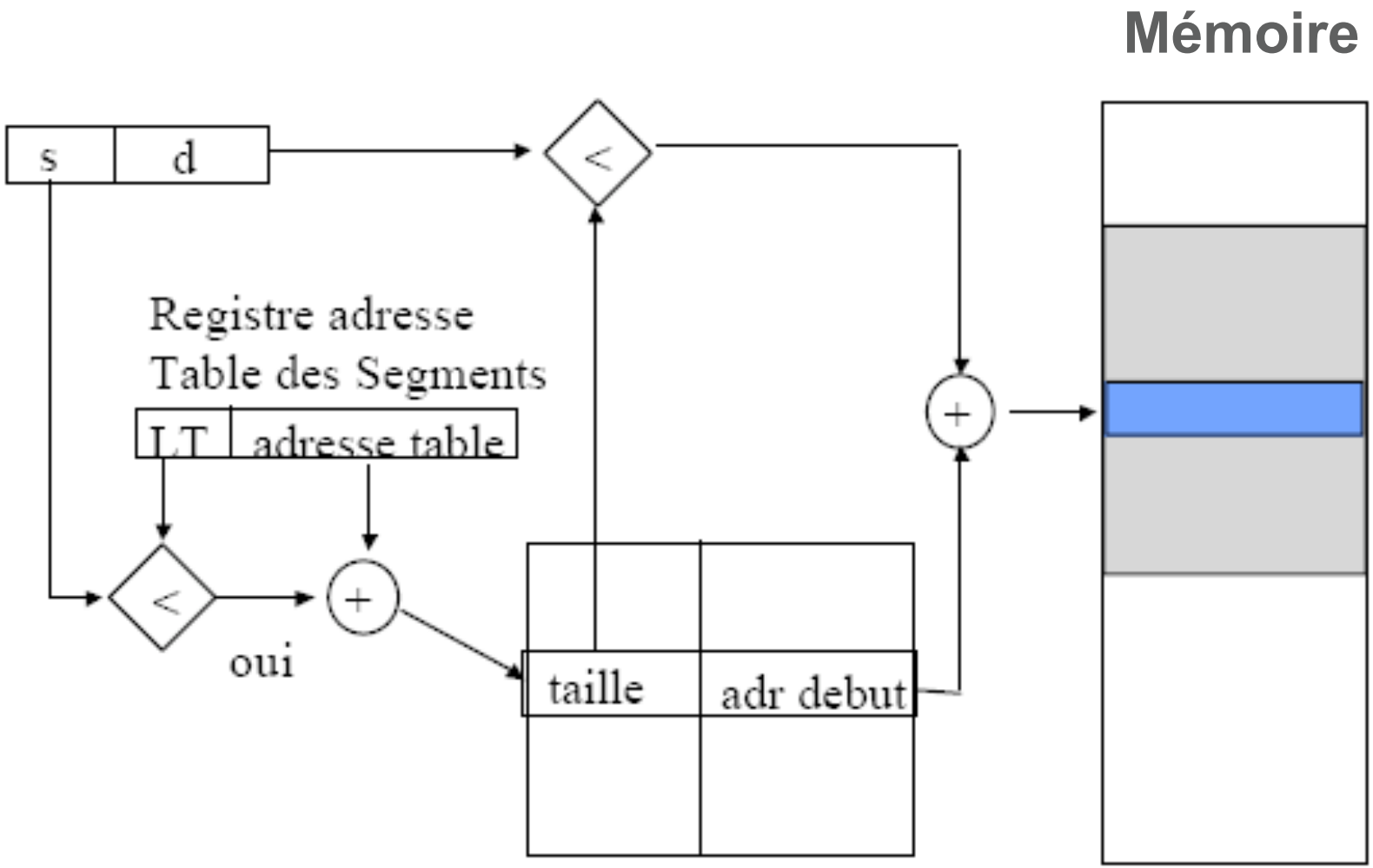


Table des segments

Segmentation

Allouer un segment S de taille Taille(S) :

→trouver une zone libre telle que
Taille (Zone Libre) >= Taille (S)

Allocations et désallocations successives des segments créent également un problème de fragmentation :

→combiner segmentation et pagination :
paginer les segments

Segmentation et pagination

La mémoire segmentée paginée

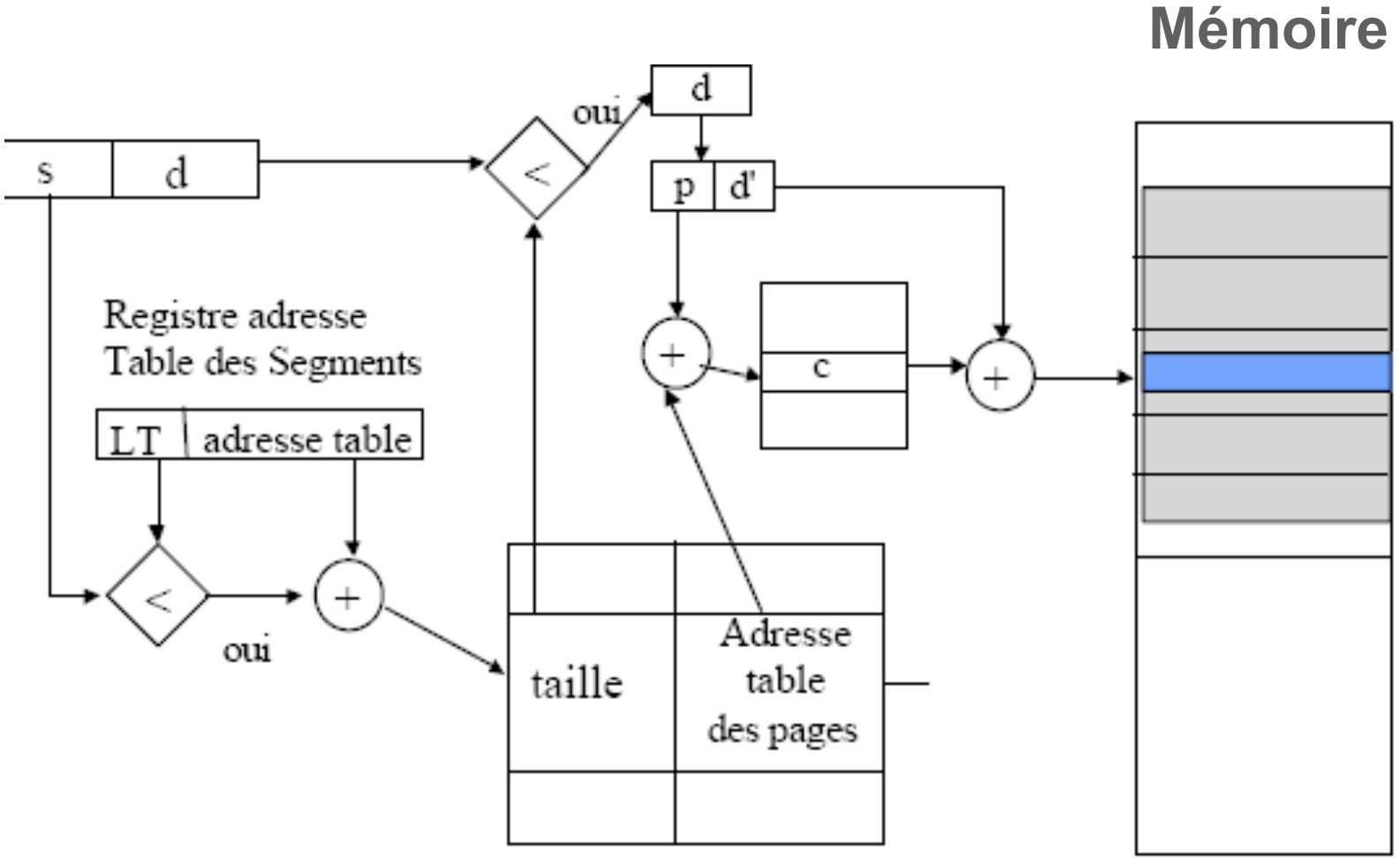


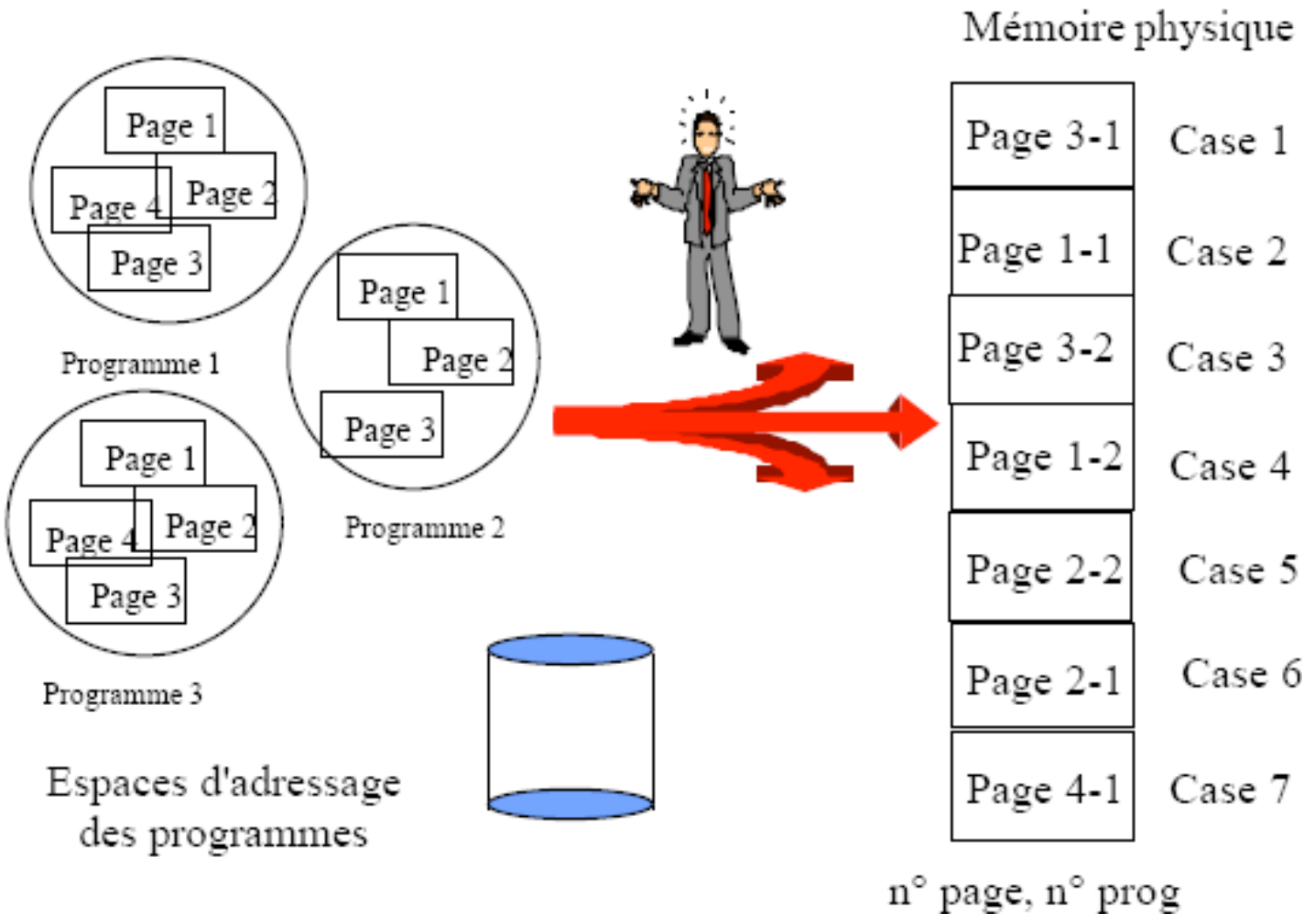
Table des segments

Segmentation et pagination

La segmentation et la pagination sont deux notions différentes qui sont utilisées conjointement.

La segmentation doit être vue comme une structuration de l'espace des adresses d'un processus, alors que la pagination doit être vue comme un moyen d'adaptation de la mémoire virtuelle à la mémoire réelle.

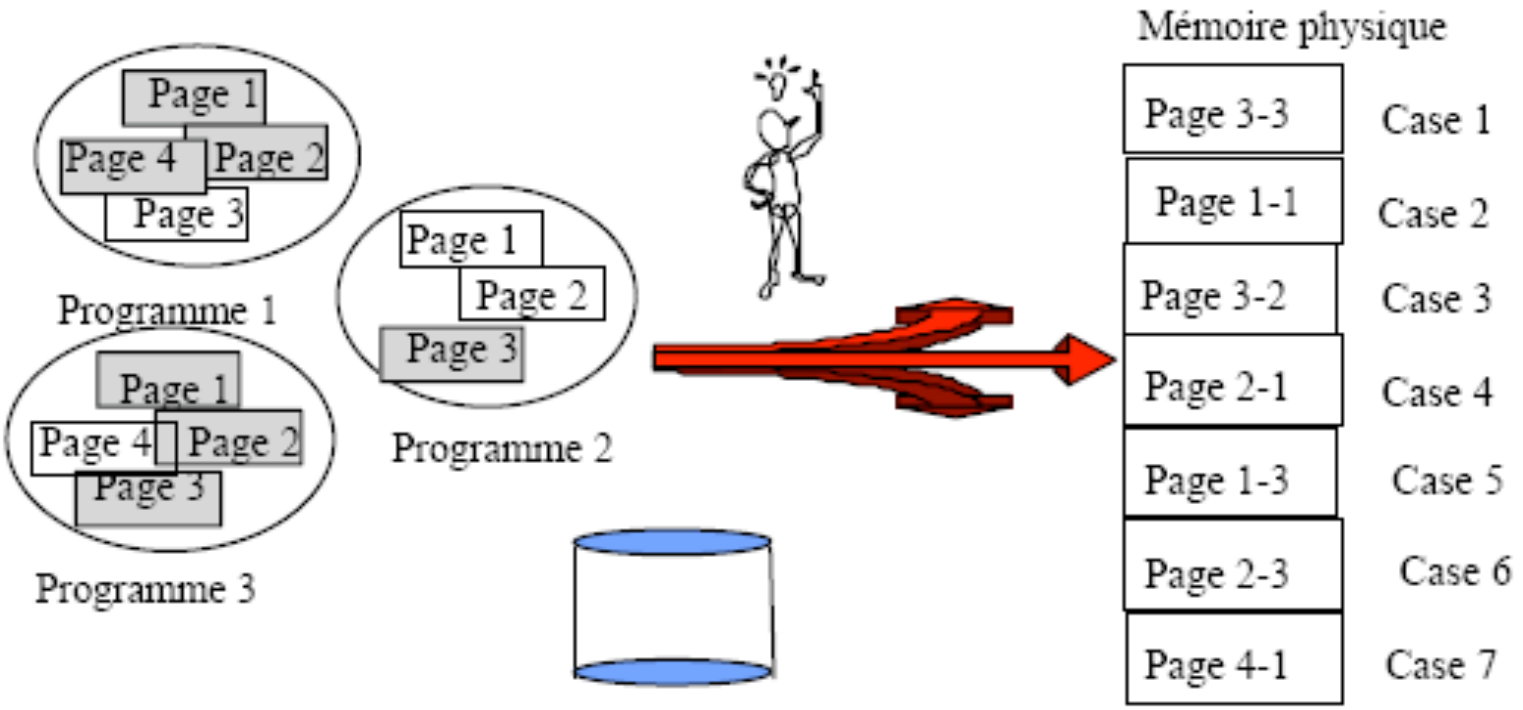
Mémoire virtuelle



Mémoire virtuelle

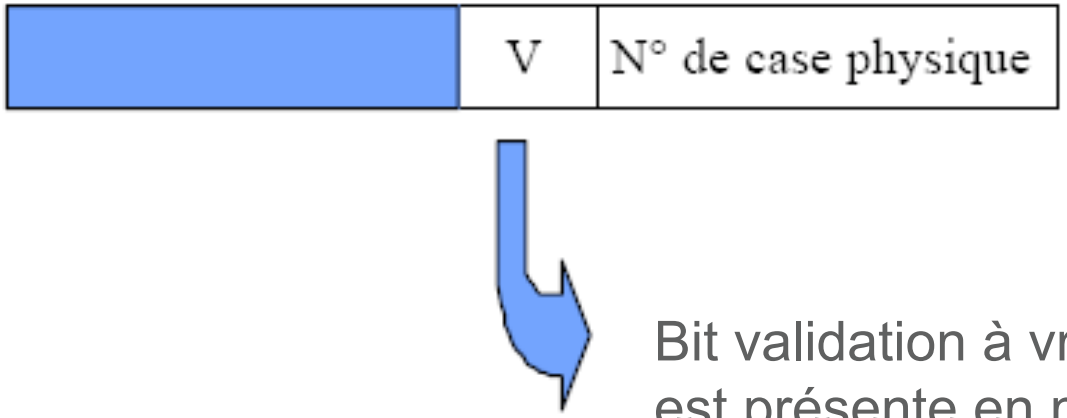
La capacité de la mémoire centrale est trop petite pour charger l'ensemble des pages des programmes utilisateurs.

→ **Ne charger que les pages utiles à un instant.**



Mémoire virtuelle

Ne charger que les pages utiles à un instant
→ il faut pouvoir tester la présence d'une page en mémoire centrale



Bit validation à vrai si la page
est présente en mémoire centrale

Organisation de la mémoire physique

Mémoire virtuelle

Bit de validation

V	2
V	4
I	-
V	7

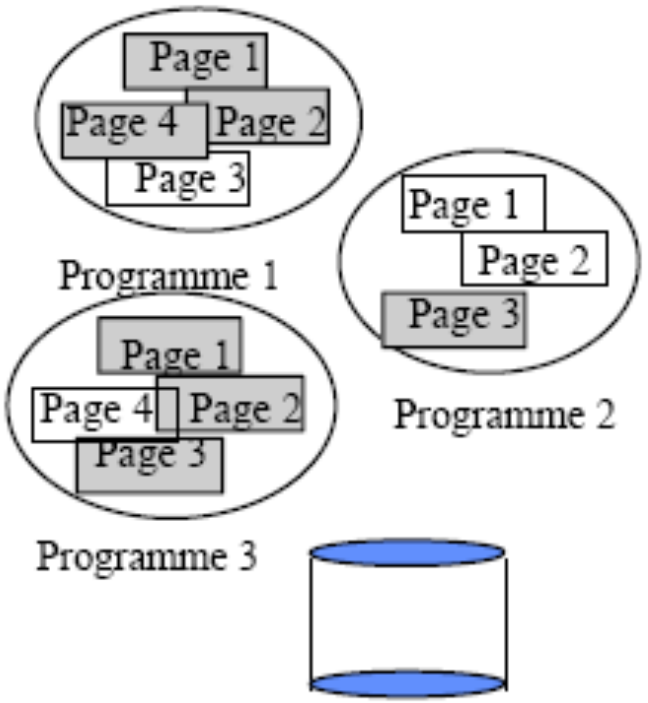
Processus 1

I	-
I	-
V	3

Processus 2

V	5
V	6
V	1
I	-

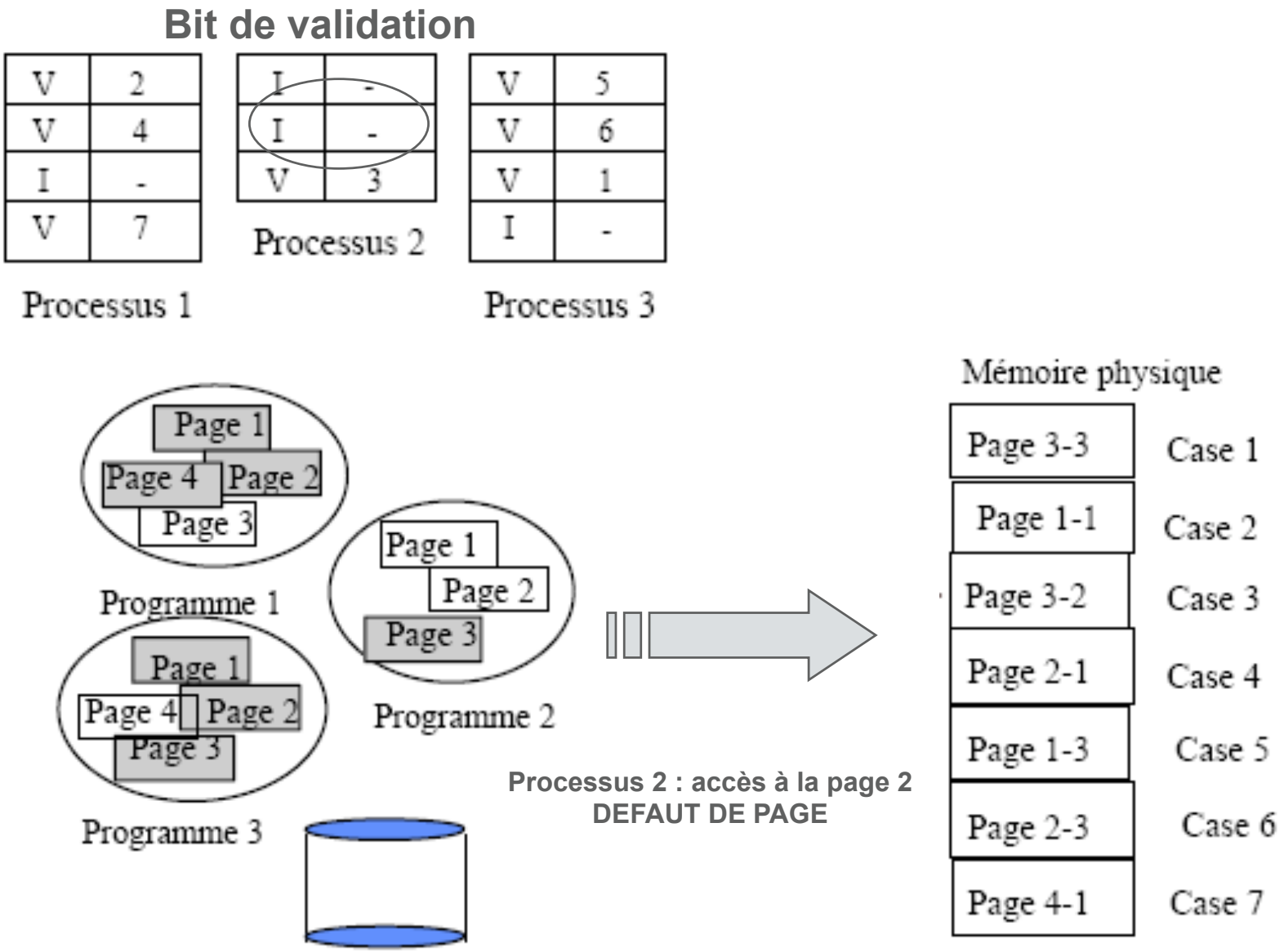
Processus 3



Mémoire physique

Page 3-3	Case 1
Page 1-1	Case 2
Page 3-2	Case 3
Page 2-1	Case 4
Page 1-3	Case 5
Page 2-3	Case 6
Page 4-1	Case 7

Mémoire virtuelle

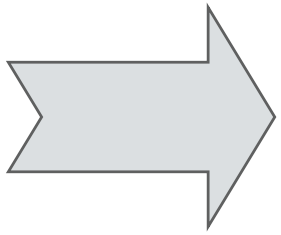


Mémoire virtuelle

Bit de validation et défaut de page

Ne charger que les pages utiles à un instant :
→ il faut pouvoir tester la présence d'une page en mémoire centrale :
→ rôle du bit de validation

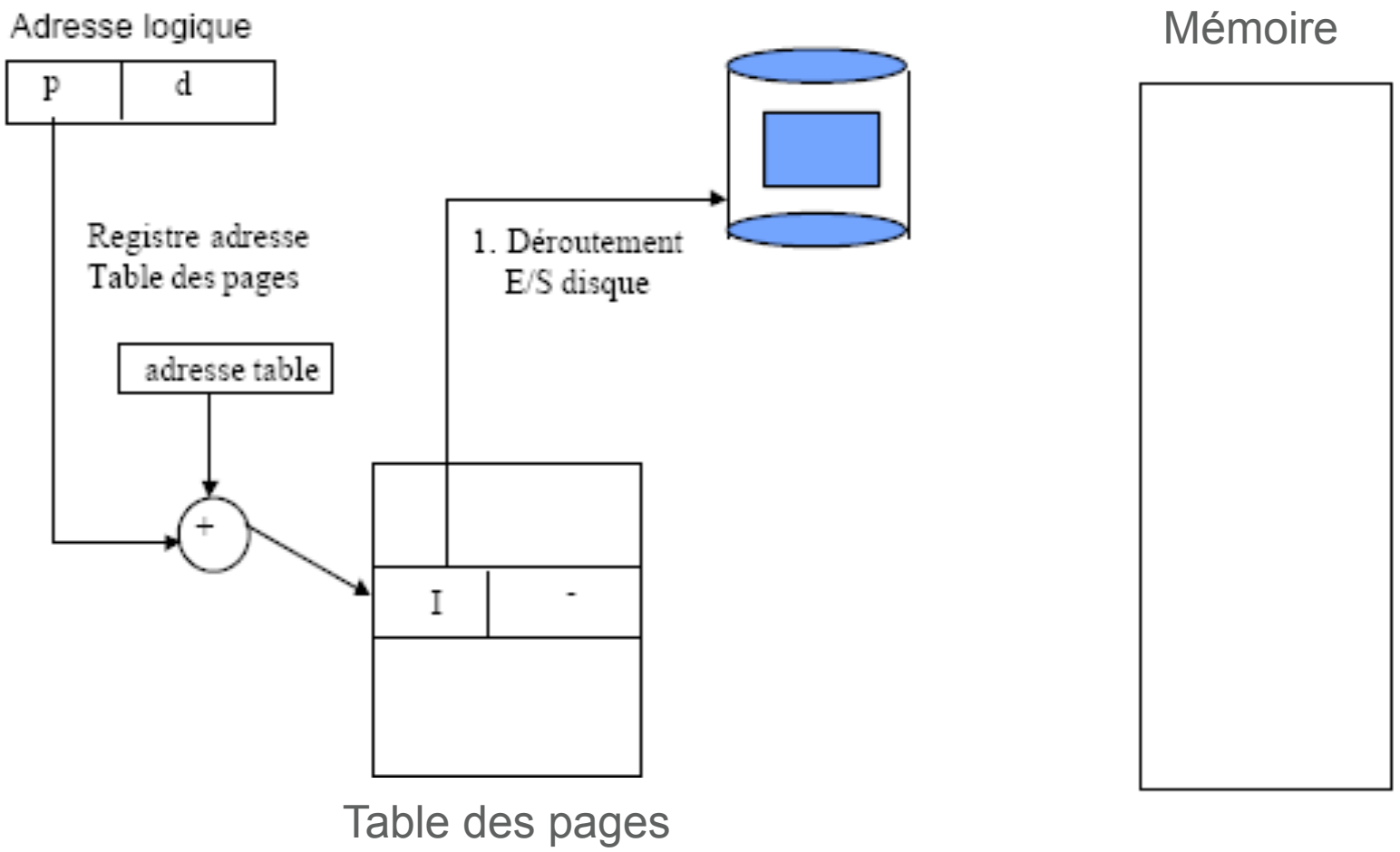
→ si un processus cherche à accéder à une page non présente en mémoire centrale, il se produit un déroutement de défaut de page



le système d'exploitation lance une entrée/sortie disque pour charger la page en mémoire dans une case libre

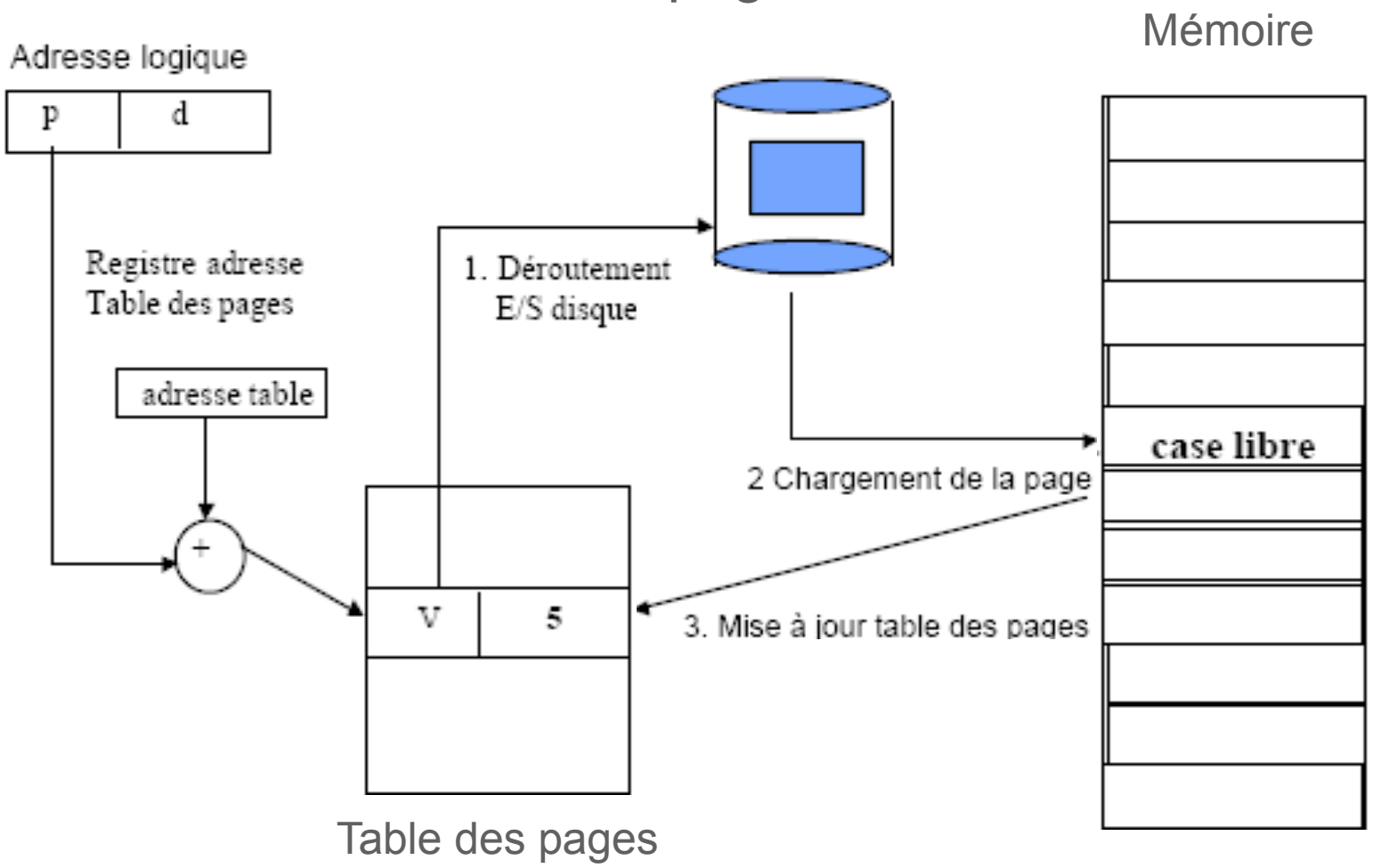
Mémoire virtuelle

Défaut de page



Mémoire virtuelle

Défaut de page



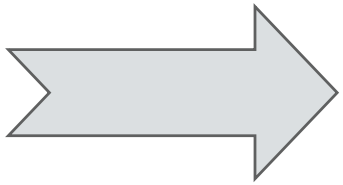
Mémoire virtuelle

Chargement de page

Lors d'un défaut de page, la page manquante est chargée dans une case libre

→ la totalité des cases de la mémoire centrale peuvent être occupées

→ il faut libérer une case globalement (parmi l'ensemble des cases) ou localement (parmi les cases occupées par les pages du processus en défaut)



le système d'exploitation utilise un algorithme pour choisir une case à libérer :

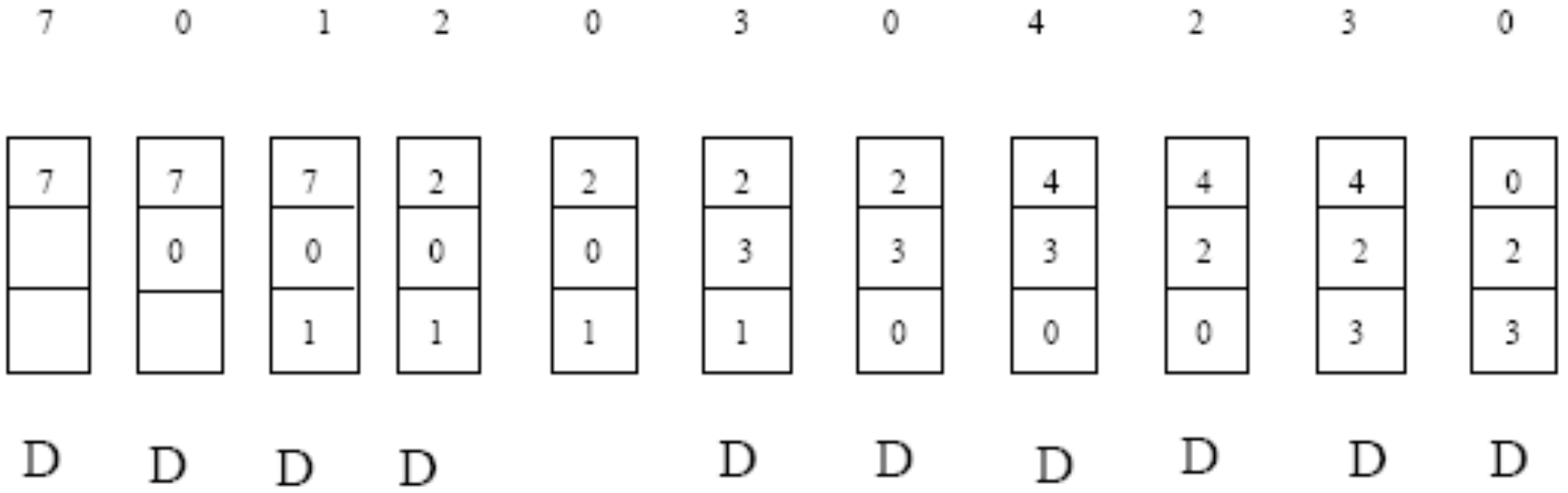
- FIFO (First In, First out)
- LRU (Least Recently Used)

Mémoire virtuelle

Algorithmes de remplacement de page

FIFO : la page la plus anciennement chargée est la page remplacée

Chaine de référence

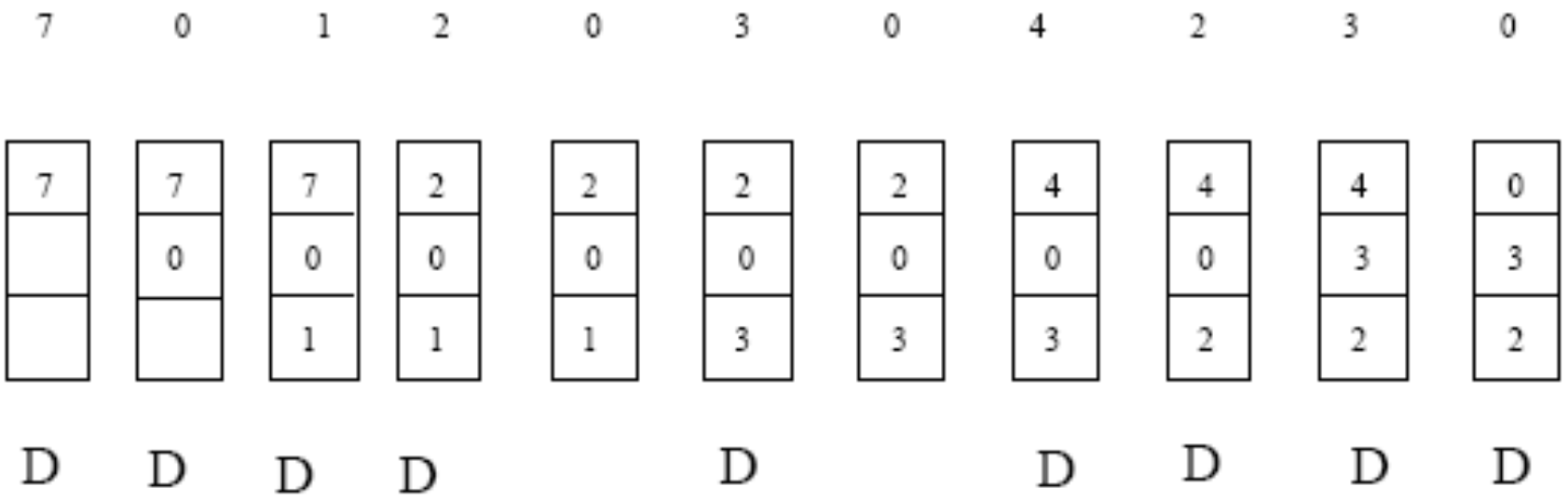


Mémoire virtuelle

Algorithmes de remplacement de page

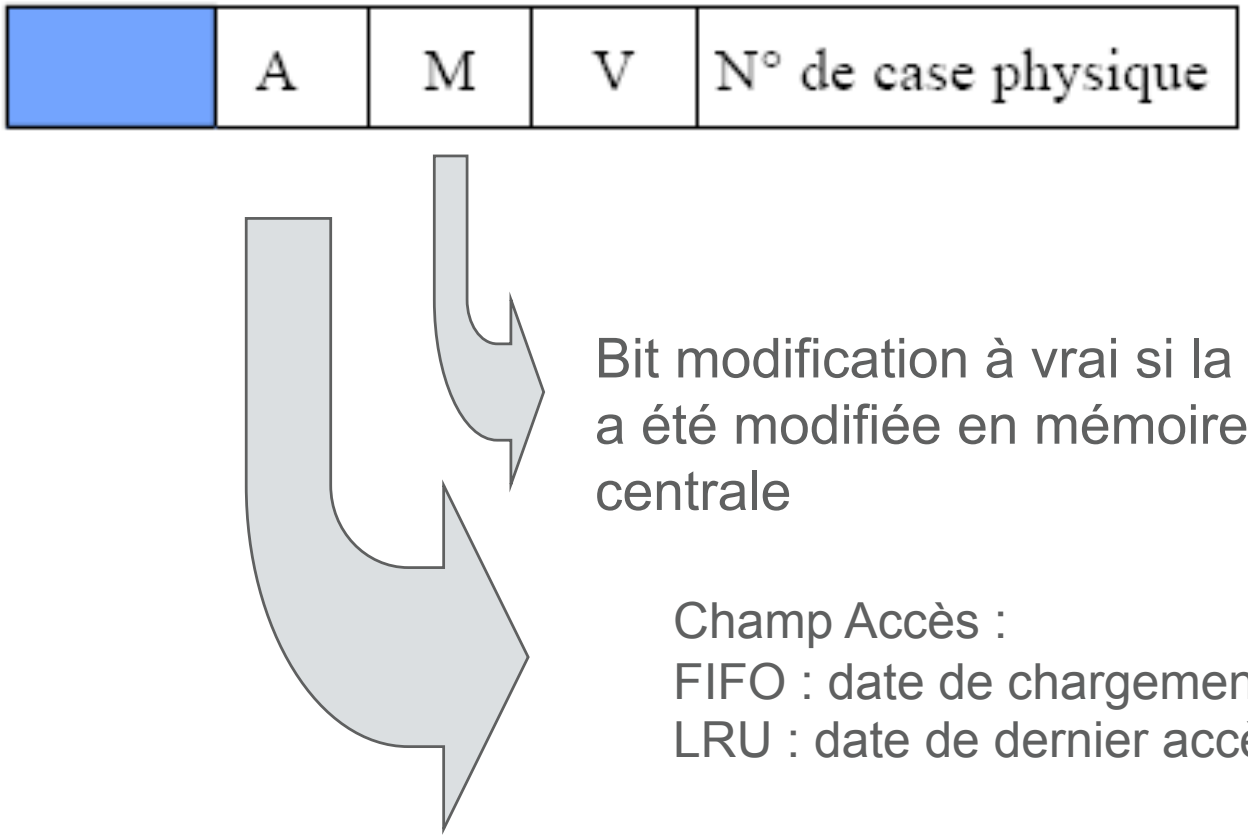
LRU : la page la moins récemment accédée est la page remplacée

Chaine de référence



Mémoire virtuelle

Ne charger que les pages utiles à un instant
→ il faut pouvoir tester la présence d'une page en mémoire centrale



Bit modification à vrai si la page a été modifiée en mémoire centrale

Champ Accès :
FIFO : date de chargement
LRU : date de dernier accès

Mémoire virtuelle

Conversion d'une adresse virtuelle

```
Procedure Conversion (in adresse_virtuelle, out adresse_physique)
debut
        entrée := adresse_virtuelle.page + adresse_table(processus)
        Si (entrée.V = FAUX)
        alors
                -- defaut de page
                Charger_page(adresse_virtuelle.page, adresse_case);
                entrée.V = vrai;
                entrée. case := adresse_case;
        fsi
        adresse physique := adresse_case + adresse_virtuelle.deplacement;
        return (adresse_physique);
fin
```

Mémoire virtuelle

Conversion d'une adresse virtuelle

```
Procedure Charger_Page (in page, out case)
debut
        Si (Trouver_case_Libre( ) = FAUX)
        alors
        Choisir_case_à_libérer (case_à_liberer, page_victime);
        si (page_victime.M = Vrai)
        alors
            Ecrire_Disque(page_victime)
        fsi
    Lire_Disque(case_à_liberer, page)
    return (case_à_ liberer)
fin
```

Mémoire virtuelle

Les pages d'un processus ne sont chargées en mémoire centrale que lorsque le processus y accède

❑ Lorsqu'un processus accède à une page non présente en mémoire centrale, il se produit un **défaut de page**.
La page manquante est alors chargée dans une case libre.

❑Si aucune case n'est libre, le système utilise un algorithme de remplacement de page pour choisir une case à libérer.

❑ L'écroulement est la situation pour laquelle un ou plusieurs processus passent plus de temps à paginer qu'à s'exécuter.