Centre Universitaire de Mila

2 ème année licence LMD Informatique

Module : Systèmes d'exploitation 1

Bessouf Hakim

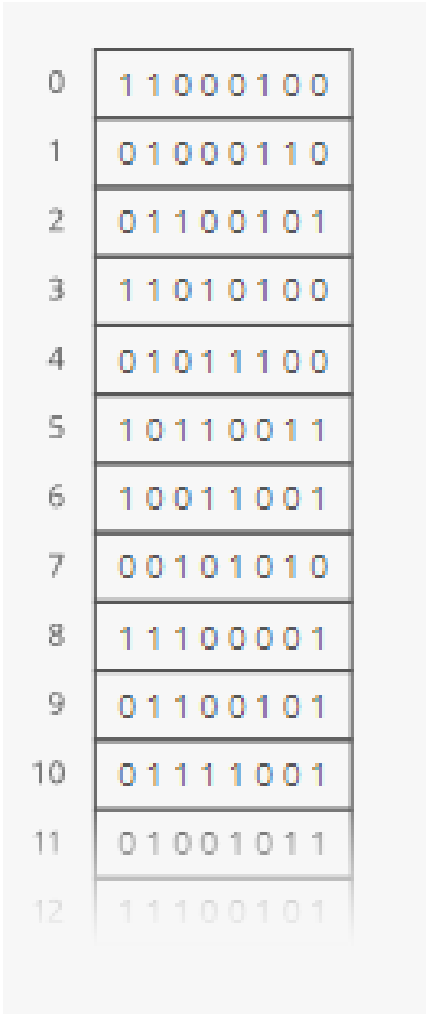
CHAPITRE 5

Gestion de la mémoire centrale

1. Objectifs d'un gestionnaire de la mémoire
2. Fonctions
3. Modes de partage de la mémoire
4. Protection de la mémoire
5. Partage de code

Introduction

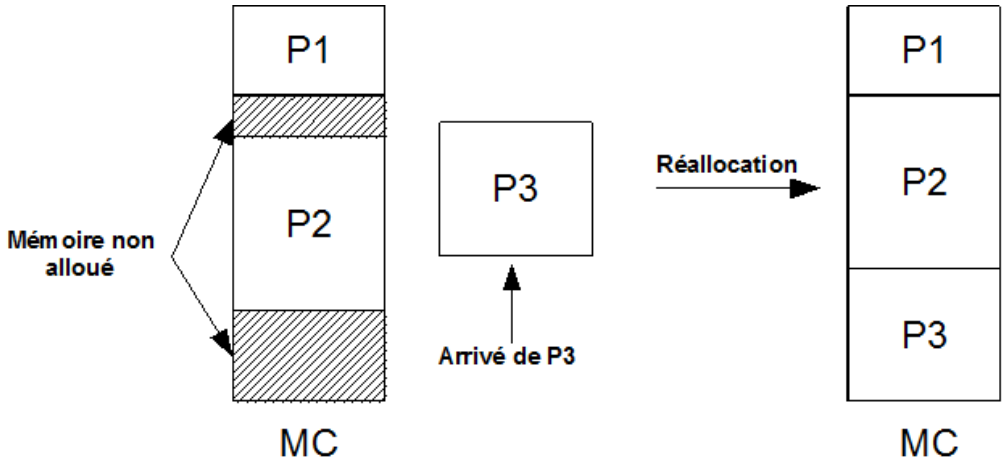
- La mémoire centrale est une partie essentielle de l'ordinateur, elle est utilisé pour stocker les programmes (donnés + codes) exécutés par le processeur.
- La mémoire centrale est découpé en plusieurs cases appelés **mots mémoire,** chaque mot mémoire a une adresse unique. Cette adresse est utilisé par le processeur pour lire et écrire les informations.



Le gestionnaire de mémoire

un module du système d'exploitation qui s'occupe de la gestion de la mémoire centrale, ces objectifs sont :

- **Le partage de la mémoire** : Le gestionnaire doit partager la MC entre plusieurs processus,
- **La protection de la mémoire** : Le gestionnaire doit pouvoir empêcher un processus d'accéder a l'espace mémoire d'un autre processus pour éviter les conflits.
- **La réallocation** : Le gestionnaire doit pouvoir réarranger (réallouer) les processus en mémoire centrale pour trouver l'espace nécessaire et charger d'autres processus.

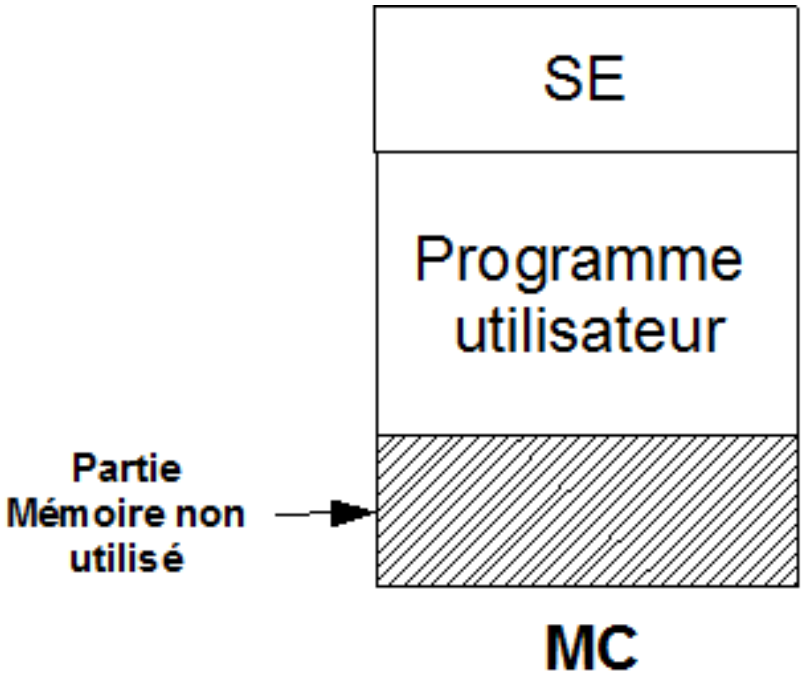


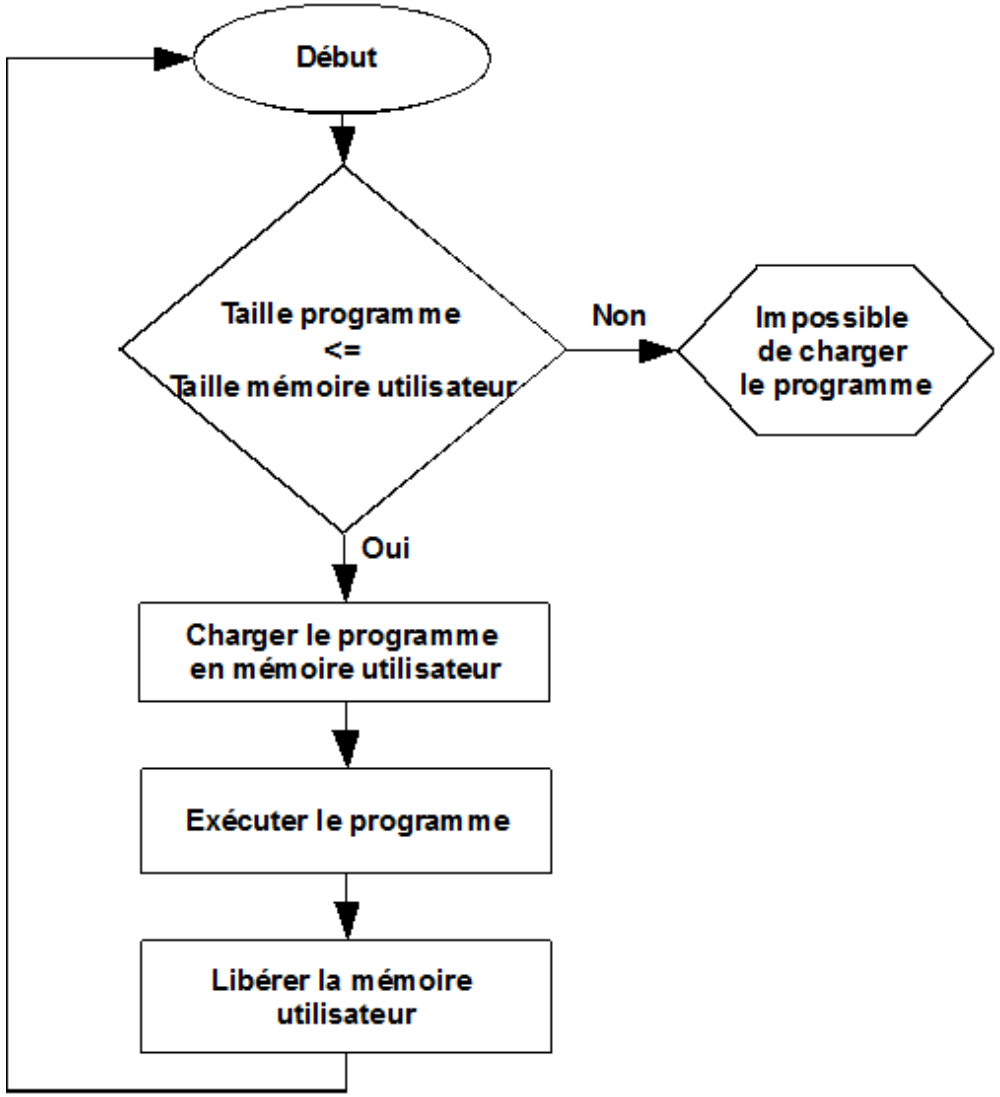
Le gestionnaire de mémoire

- **Allouer l'espace mémoire aux processus** : Le gestionnaire de mémoire doit déterminer une technique d'allocation de la mémoire.
- **Libérer l'espace mémoire** : Le gestionnaire de mémoire doit déterminer une technique pour libérer l'espace mémoire quand un processus termine sont exécution.
- **Déterminer les parties libres** : Le gestionnaire de mémoire doit garder une trace de chaque partie de la mémoire centrale pour pouvoir la gérer correctement.

Modes de partage de la mémoire (stratégies d'allocation de la mémoire)

- **Une seule zone contiguë (single continuous allocation)**
  Dans cette stratégie un seul programme est exécuté a la fois (**monoprogrammation**). La mémoire centrale est divisé en deux partie, la première partie contient le système d'exploitation et la deuxième partie contient le programme utilisateur.





Organigramme de l'allocation de la mémoire avec une seule zone contiguë

RQ: Cette stratégie d'allocation est inefficace car une partie de la mémoire peu être non utilisé.

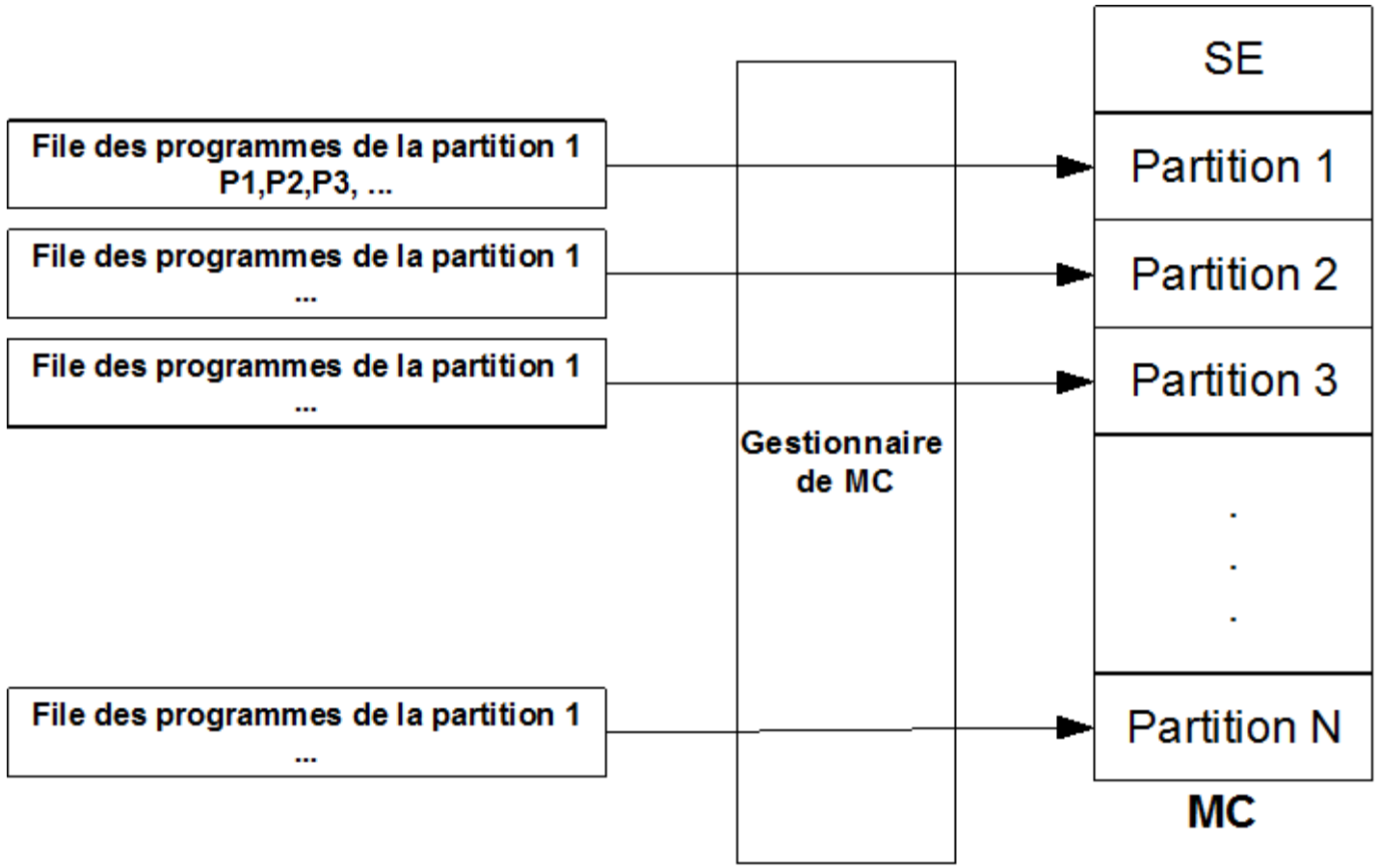
Modes de partage de la mémoire (stratégies d'allocation de la mémoire)

- **Partitions multiples**
  Dans cette stratégie la MC est divisé en plusieurs partitions, chaque partition peut contenir un processus. On peut donc avoir plusieurs processus en mémoire centrale en même temps (**multiprogrammation**).

1. **Partitions multiples statiques**
2. **Partitions multiples dynamiques**

1. Partitions multiples statiques

- Dans ce schéma la taille et le nombre de partitions est fixé a l'avance lors du démarrage du système d'exploitation. Pour exécuter un programme il y a deux cas passibles :
- **Le programme à exécuter est en code absolus :**
  Dans ce cas les adresses du programme sont des adresses physiques et le programme est donc lié a une partition précise et ne peut pas s'exécuter dans une autre partition.
- Pour pouvoir exécuter les programmes le gestionnaire de mémoire centrale associe a chaque partition une file des processus en attente d'exécution comme décrit dans la figure suivante :

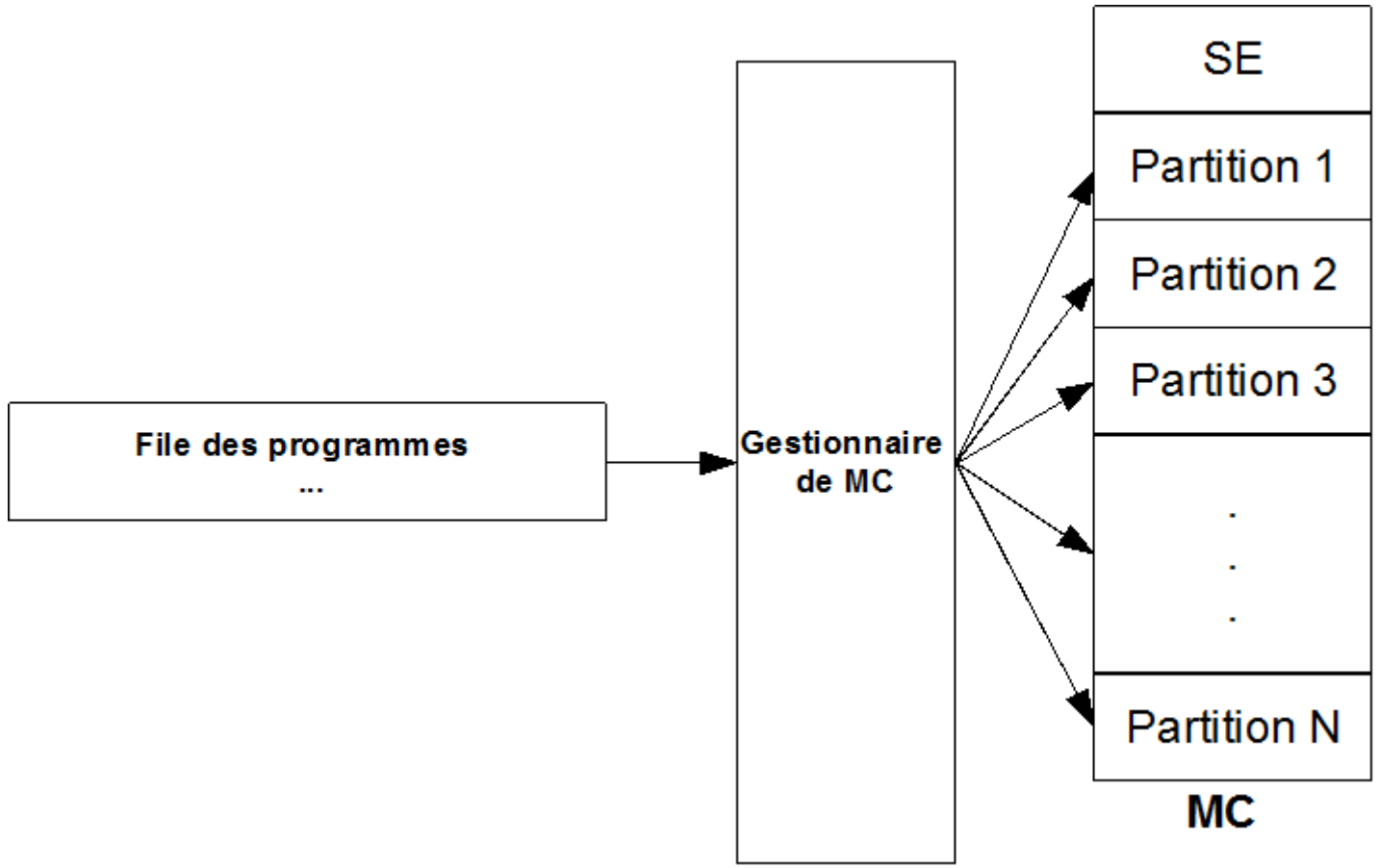


RQ: L'inconvénient du chargement absolus est que des partitions peuvent être libres alors que les programmes sont dans une file d'une autre partition.

Une files de programmes associé a chaque partition statiques.

1. Partitions multiples statiques

- **Le programme à exécuter est en code relogeable**
  Dans ce cas les adresses du programme sont des adresses logiques (non absolus) et le programme peut être chargé et exécuté dans n'importe quelle partition qui a une taille suffisante.
  Le gestionnaire de mémoire associe a tous les processus une seule file d'attente.
- Pour exécuter un programme le gestionnaire de MC doit trouver une partition de taille suffisante qui est libre et charger le programme dans cette partition
- le gestionnaire doit aussi convertir les adresses logique en adresses physique suivant la partition utilisé.
- Pour convertir les adresses logiques en adresses physique on ajoute l'adresse de début de la partition a toutes les adresses logiques du programme.

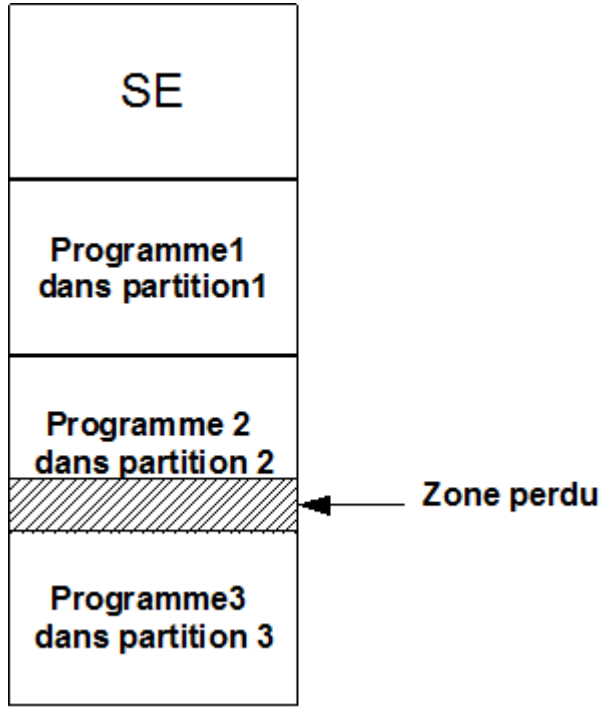


Une seule file de programmes est associé a toutes les partitions statiques.

1. Partitions multiples statiques

- **Problème de fragmentation**
  La fragmentation interne :

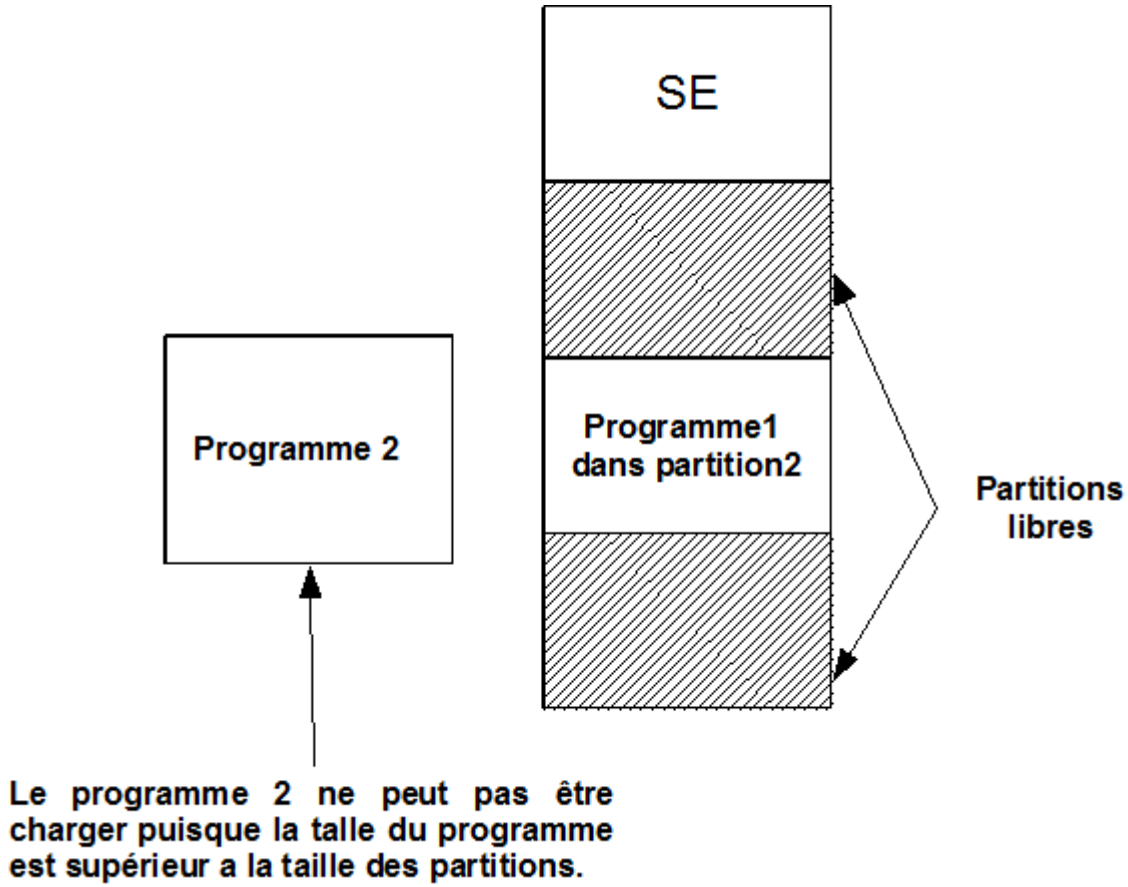
quand un programme est chargé dans une partition et la taille du programme est inférieure a la taille de la partition on aura de la mémoire non utilisé, on dit qu'il y a une fragmentation interne.



1. Partitions multiples statiques

- **Problème de fragmentation**
  La fragmentation externe :

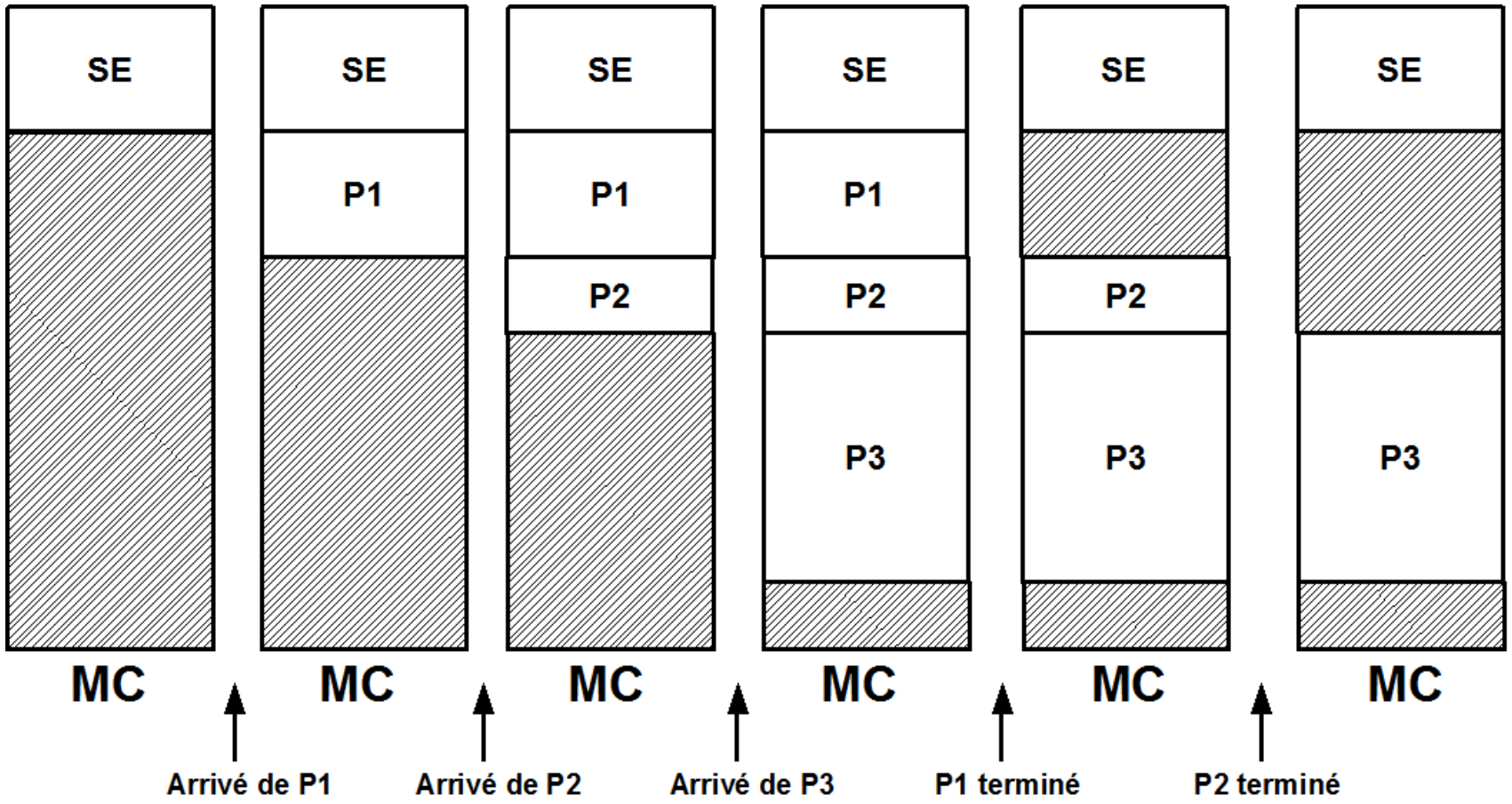
Si la taille d'un programme est supérieur a la taille de toutes les partitions libres et inferieur à la somme des partitions libres, donc le programme ne peut pas être chargé. Dans ce cas on dit qu‘on a une fragmentation externe.



2. Partitions multiples dynamiques

- Le gaspillage de la mémoire centrale dû a la fragmentation dans le partitionnement multiple statique a conduit au schéma de partitionnement multiple dynamique.
- Dans ce schéma suivant on partitionne la mémoire centrale dynamiquement selon la demande, et on alloue aux programmes des partitions exactement égale a leurs tailles. Lorsque un programme termine son exécution sa partition est récupéré pour être allouer a un autre programme. Si une partition libre est adjacente a une autre partition libre, les deux partitions sont **compactés** en une seule partition plus grande.

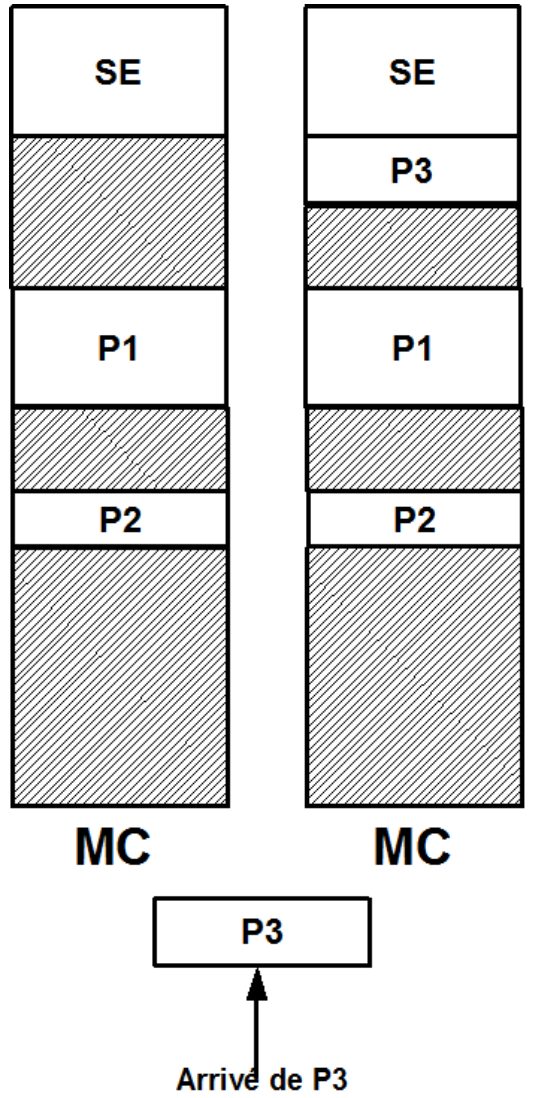
2. Partitions multiples dynamiques



2. Partitions multiples dynamiques

- Le placement des programmes dans les partitions se fait suivant différentes stratégies: **Stratégie du premier qui convient (first fit)**:

Dans cette stratégie le gestionnaire de mémoire place le programme dans la première partition libre qui a une taille suffisante pour l'exécution du programme.

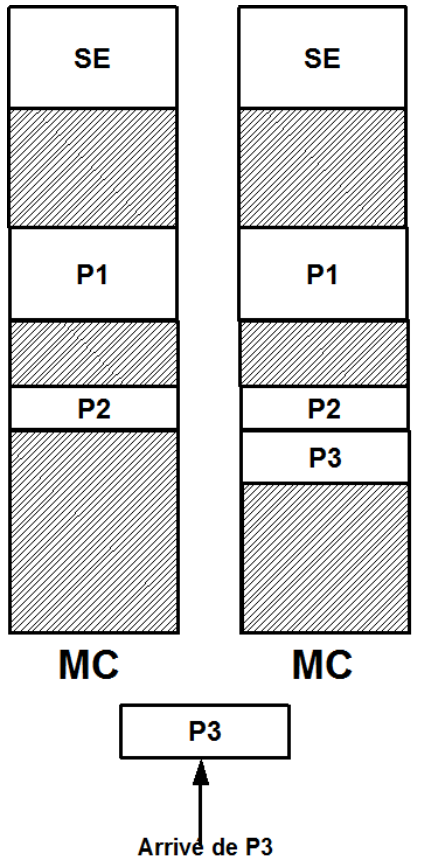


2. Partitions multiples dynamiques

- **Stratégie du pire qui convient (Worst fit):**
  Dans cette stratégie le gestionnaire de mémoire place le programme dans la partition libre la plus grande qui a une taille suffisante pour l'exécution du programme.

La stratégie du **premier qui convient** est **rapide** mais conduit a des **pertes de mémoire** importantes du a la fragmentation. Les stratégies du **meilleur qui convient** et du **pire qui convient** sont **moins rapides** puisque il nécessites la recherche dans toutes les partitions libres.

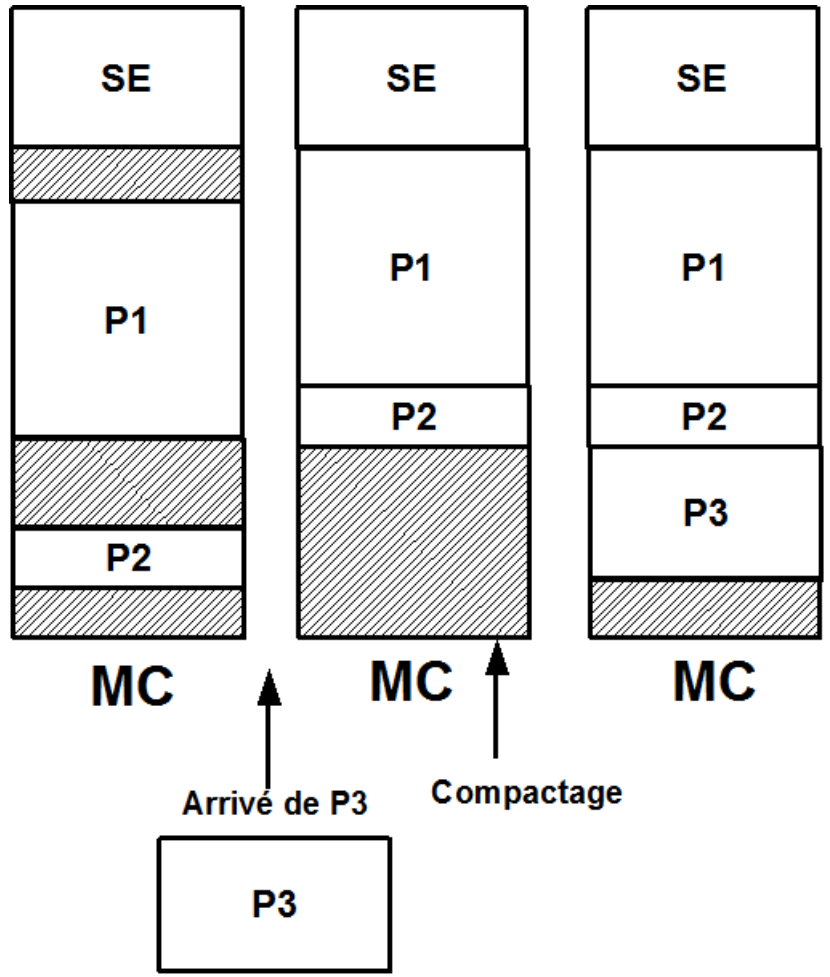
RQ: Des simulation ont montrés que la stratégie du premier qui convient est meilleur que la stratégie du meilleur qui convient et les deux stratégies sont meilleurs que la stratégie du pire qui convient.



2. Partitions multiples dynamiques

- **Le compactage de la mémoire:**
- Quelque soit la stratégie d'allocation utilisé il y a toujours une fragmentation externe.
- Le compactage consiste a regrouper toutes les partitions mémoires non utilisés en une seule partition plus grande dans la quelle on peut charger les programmes.

- **RQ:** *l'opération de compactage est très coûteuse en temps, si par exemple on a une machine avec 1 Go de RAM et qui peut copier 4 octets en 20 ns, on aura besoin de 5 secondes pour compacter toute la mémoire*

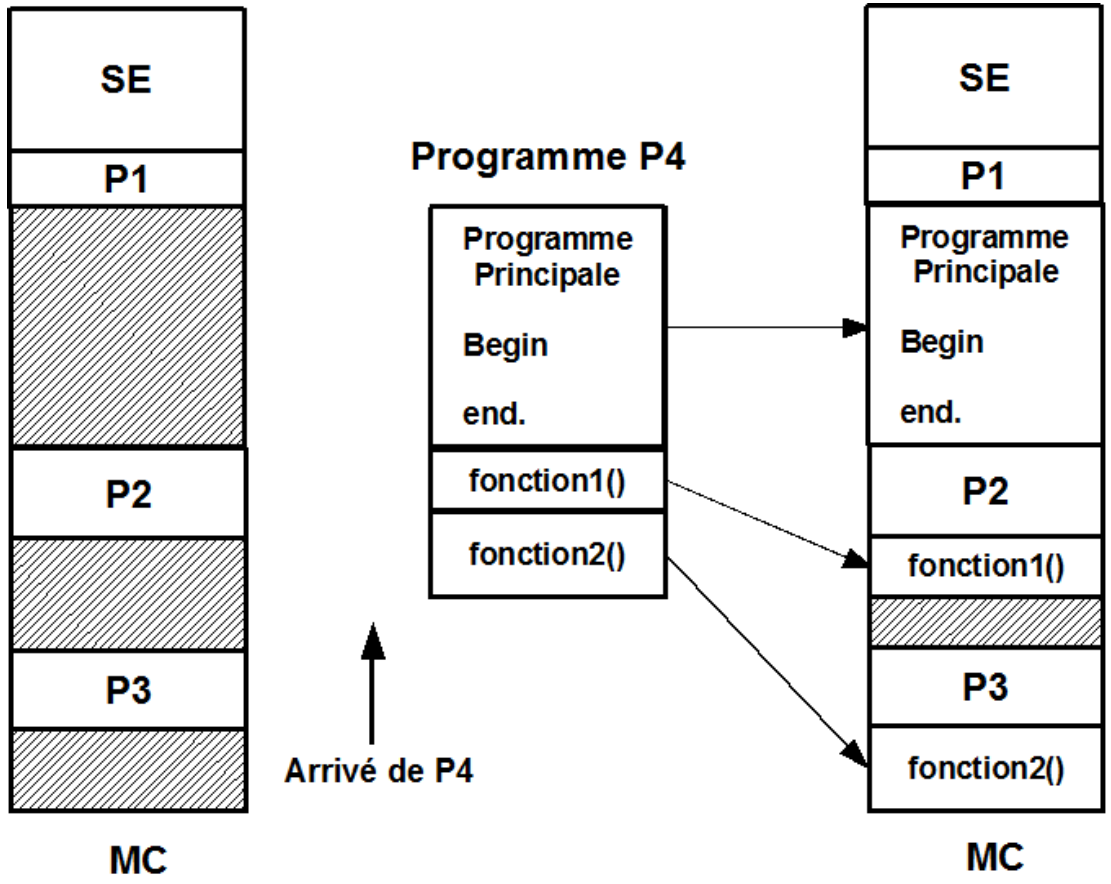


Modes de partage de la mémoire (stratégies d'allocation de la mémoire)

- **la segmentation:**

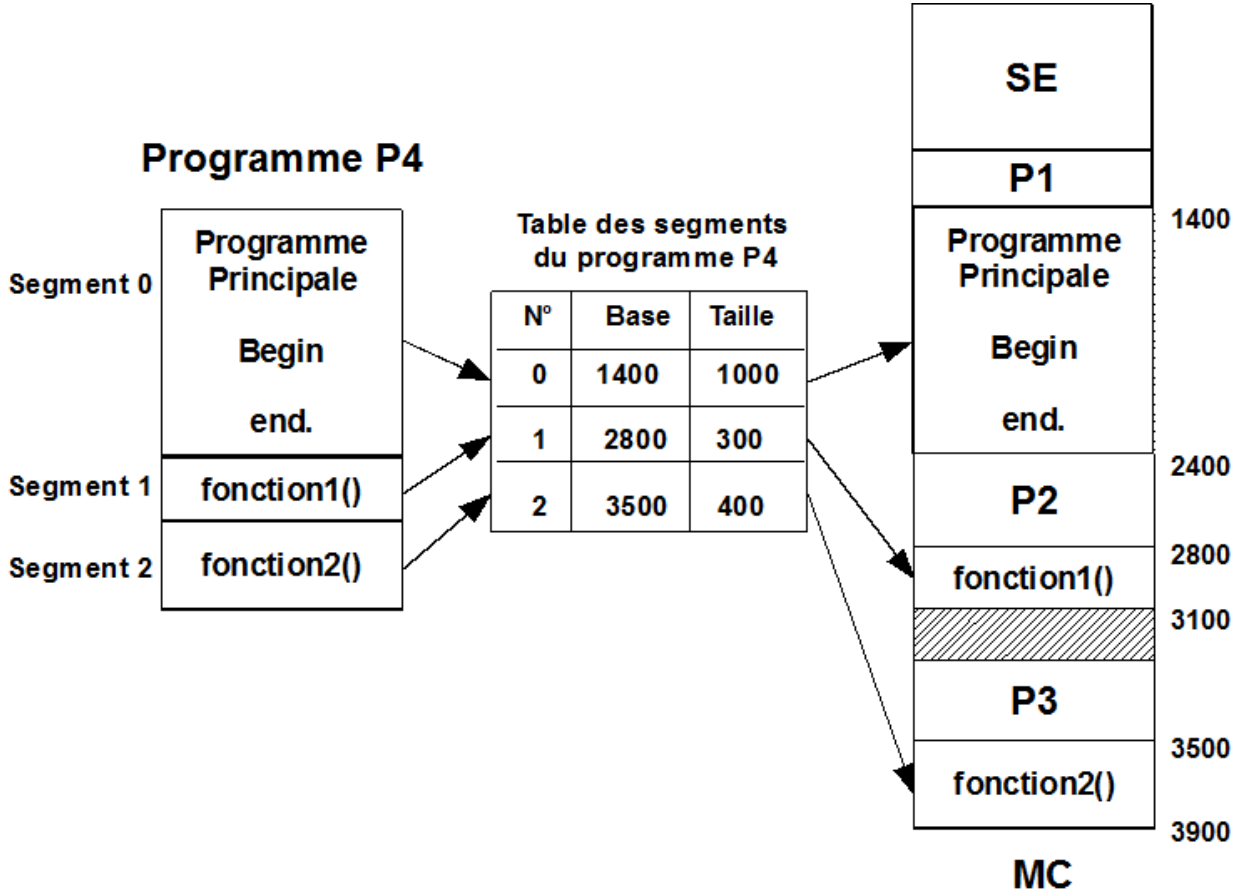
on divise le programme en plusieurs segments, chaque segment correspond a une entité logique (programme principale, procédures et fonctions, structures de données… etc). Un **compilateur pascal** par exemple produit des segments différents pour :

- **Les variables globales,**
- **La pile d'appels de procédures,**
- **Le code de chaque procédure ou fonction,**
- **les variables locales de chaque fonction.**

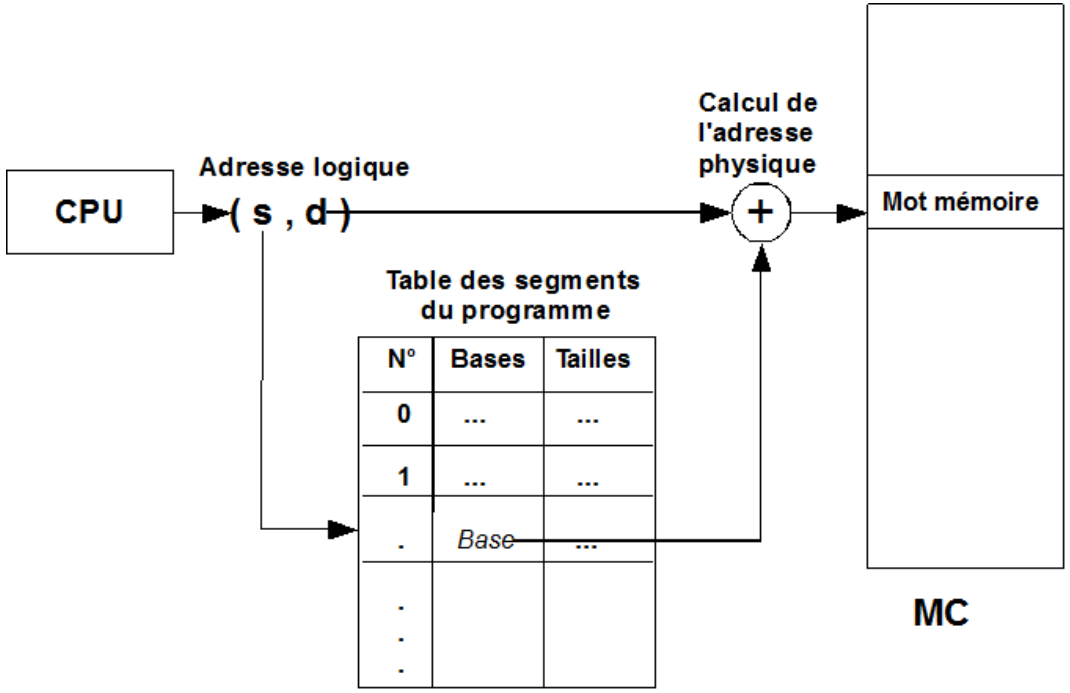


A l'intérieur d'un segment **les adresses sont relatives** au début des segments (**ils commences de 0** jusqu'à la taille du segment).
Les segments d'un même programme peuvent être **chargés dans des espaces mémoire séparés**, la gestion de la mémoire centrale est donc **plus efficace**.

A chaque programme en exécution on associe une **table de segments** qui pour chaque segment du programme donne la **base** (adresse de début) et la **taille** du segment en MC.



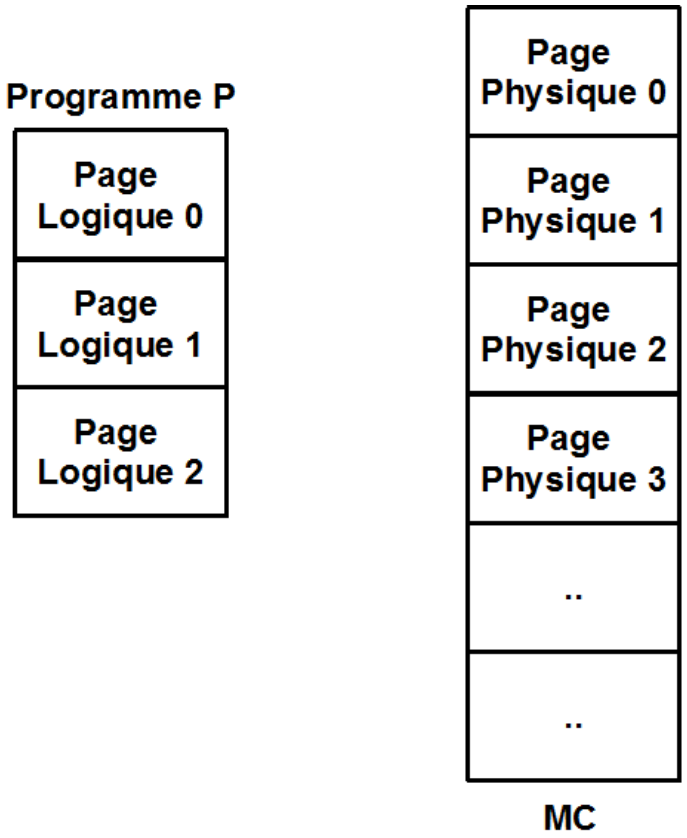
- Les adresses du programme sont des **adresses logiques**, une adresse logique spécifie le segment et le déplacement dans le segment. La correspondance entre l'adresse logique du programme et l'adresse physique en mémoire centrale est réalisé en utilisant la table de segments du programme comme suit :
  L'adresse logique est composé du numéro du segments **s**, et du déplacement dans le segments **d**. Adresse logique = <**s,d**>
  Le numéro du segment **s** est utilisé comme indexe dans la table des segments pour trouver l'adresse de base du segment en mémoire centrale, et calculer l'adresse physique.



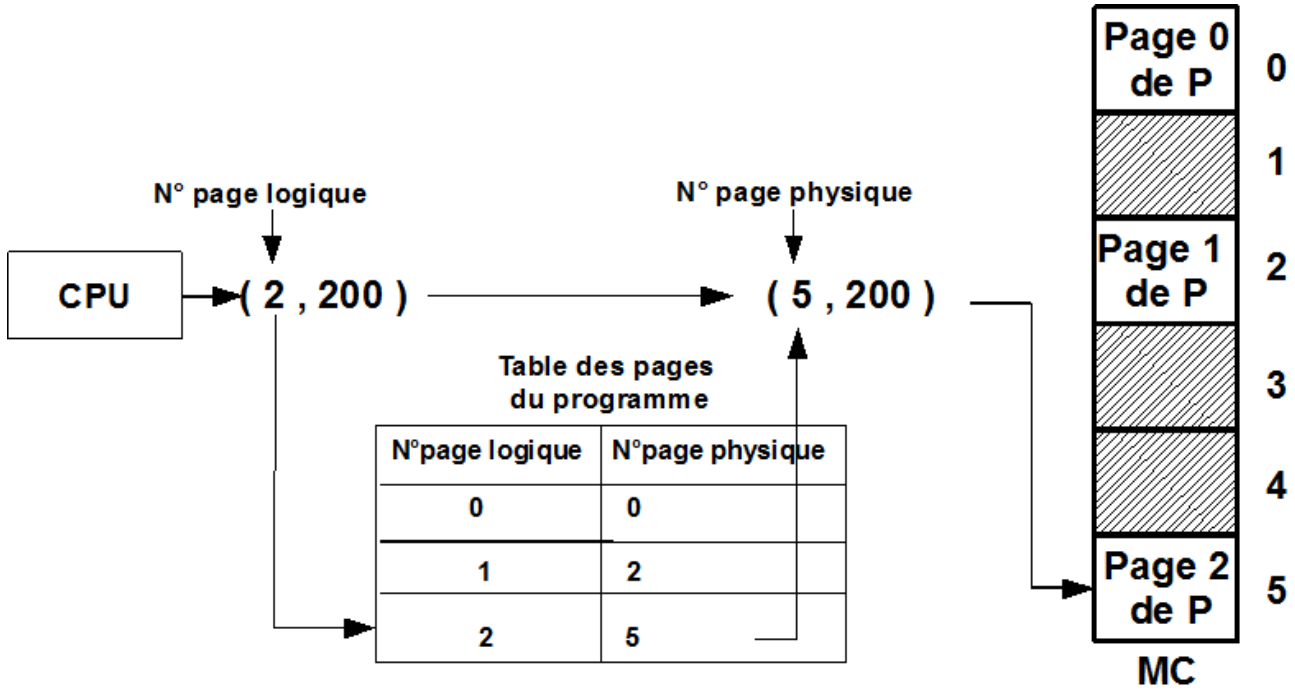
RQ: *Le numéro de segment S doit être inférieur a la longueur de la table des segments, aussi le déplacement dans le segment doit être compris entre 0 et la taille du segment. Dans le cas contraire on aura une erreur d'adressage*

Modes de partage de la mémoire (stratégies d'allocation de la mémoire)

- **La Pagination**
  Dans la pagination la MC est divisé en plusieurs partitions de taille égales et fixes appelées pages physiques. Aussi l'espace d'adressage des programmes est divisé en plusieurs partitions appelés pages logiques. La taille des pages logiques est égale a la taille des pages physiques.
- Un programme peut être chargé dans des pages physiques non contigus, ce qui permet d'éliminer le problème de la fragmentation externe.



- A chaque programme en exécution on associe une **table des pages** qui fait la correspondance entre les pages logiques et les pages physiques en MC .
- Chaque adresse généré par le processeur pendant l'exécution d'un programme est divisé en deux parties , un **numéro de page logique p ,** et **un déplacement d**.



RQ: *La pagination* ***élimine le problème de la fragmentation externe*** *mais le problème de la* ***fragmentation interne*** *reste toujours présent*

Le Swapping

Des fois la taille de la mémoire centrale ne suffit pas pour charger tous les programmes en attente d'exécution. Une solution consiste a sauvegarder temporairement dans une mémoire secondaire (en générale le disque dur) les programme qui sont bloqués (en attente d' E/S ou d'un événement) et charger d'autres programmes dans les partitions
libérés. L'opération est réalisé comme suit :
- Un programme est chargé dans sa totalité en MC (**Swap-in**),
- Le programme restera dans la MC jusqu'à sa terminaison ou son blocage,
- En cas de blocage, le programme peut être transféré en mémoire secondaire (**Swap-out**),
- Un ou plusieurs nouveaux programmes sont chargés dans la partition libéré.

Le swapping est très coûteux en temps a cause du temps de transfère des programmes des mémoires secondaires.