CHAPITRE IV: GESTION DE LA MÉMOIRE CENTRALE

I/ Généralités :

Une mémoire est un système capable d'acquérir, de conserver, et de restituer des informations binaires dans un ordinateur. La capacité d'une mémoire est la quantité d'information qu'elle peut stocker, elle est exprimée en bits ou en mots de 2^n bits.

Le temps d'accès et le temps qui s'écoule entre la demande de l'information et le moment où elle est disponible.

Quatre types de mémoires qui se diffèrent selon l'utilisation, la capacité, le temps d'accès et le coût :

- Les registres (sont les plus petites mémoires qui ont le plus court temps d'accès et les plus chers)
- La mémoire cache (elle a une faible capacité, très rapide, et le coût très élevé).
- La mémoire centrale (capacité de 128 MO → 2 GO, avec un accès très rapide, moins cher).
- Les mémoires auxiliaires (elles ont une très grande capacité, avec un accès long et sont les moins chers).

II/ Définition de la mémoire centrale :

La mémoire centrale est un ensemble d'emplacements ou mots mémoires, chacun possédant sa propre adresse, elle répond aux opérations de lecture et d'écriture. La MC est utilisée pour le stockage des données et des programmes au cours d'exécution, c'est l'intermédiaire entre le processeur et le reste des organes de l'ordinateur.

L'ensemble de programmes permettant la gestion et le contrôle d'utilisation de la mémoire centrale est dit système de gestion de mémoire centrale.

III/ Objectifs du gestionnaire de mémoire :

Les objectifs du gestionnaire de MC sont :

1/ Le partage et le contrôle de l'espace mémoire entre les processus.
2/ La protection de l'espace alloué à chaque processus.
3/ La réallocation et le réarrangement de la mémoire pour optimiser son utilisation.
4/ L'organisation logique, c'est-à-dire, division logique de l'espace en partitions, pages ou segments.
Pour cela le gestionnaire de la MC :

- Alloue l'espace mémoire aux processus.
- Libère l'espace occupé.
- Garde trace des parties mémoire occupées et libres.

- Gère les opérations de Swapping entre la MC et la mémoire secondaire.

IV/ Différents modes de partage de la MC :

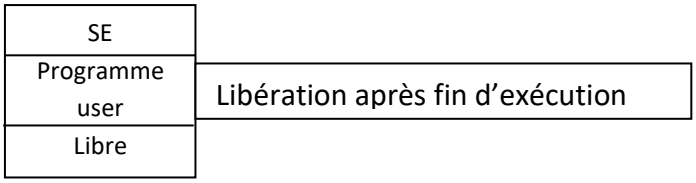
1/ **Modes n'utilisant pas la mémoire virtuelle :**

Les programmes sont chargés seulement dans la MC, parmi ces modes on peut citer :

a/ Zone unique contiqüe : utilisé dans les systèmes séquentiels, où il existe un seul programme à la fois d'une façon à partager la MC entre ce programme et le système d'exploitation.

Inconvénients :

- Mauvaise utilisation de la mémoire.
- Les tailles des programmes sont limitées à cette zone.



N.B : Ce schéma est utilisé dans les premiers mini-ordinateurs (IBM. DS/360)

b/ Schéma à partitions multiples : appliqué dans l'exécution de plusieurs programmes (multiprogrammation) → c'est de diviser l'espace en **n** partitions, on distingue deux stratégies :

- Partitions multiples fixes :

La MC est divisée en **n** partitions de tailles fixées à la génération du SE, ce partitionnement ne se change pas jusqu'à l'arrêt de la machine. Un algorithme d'allocation est chargé de chercher une taille suffisante au programme en attente.

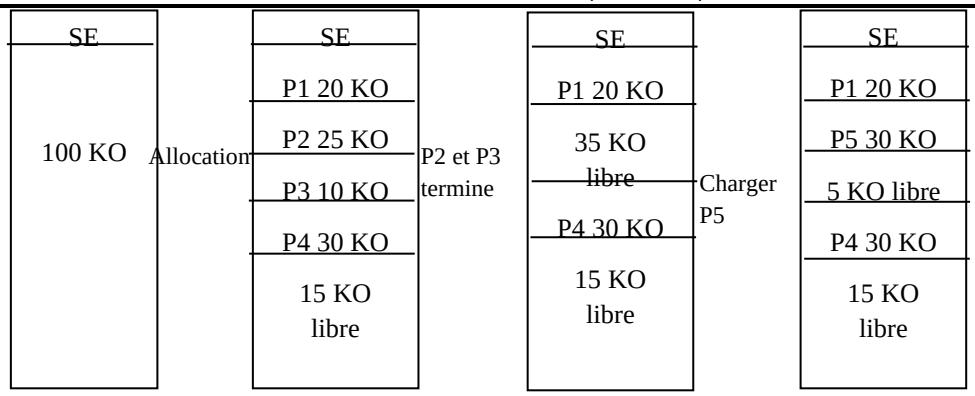
Inconvénients :

- Apparence de plusieurs fragments dans la mémoire inutilisable.
- Des partitions peuvent être libres alors que des programmes sont en attente → taille de programme > taille de la partition.
- Problème avec les programmes absolus qui nécessitent des adresses de chargement absolues → où on ne peut pas utiliser les partitions libres.

- Partitions multiples variables :

La mémoire est découpée dynamiquement selon la demande des programmes.

Exemple : partition 100KO pour programme utilisateur, P1 = 20, P2 = 25, P3 = 10, P4 = 30, P5 = 30



L'allocation dans cette stratégie est faite selon l'un des algorithmes suivants :

- *First Fit* : Le programme est placé dans la première partition trouvée >= à celle du programme. Elle a comme inconvénient de laisser plusieurs fragments inutilisables.
- *Best Fit* : cette stratégie cherche de trouver la meilleure partition dont le résidu soit minimal (taille de partition – taille du programme <<<).
- *Worst Fit* : le programme est placé dans la partition la plus large, son objectif est de gagner une nouvelle partition pour un autre programme.

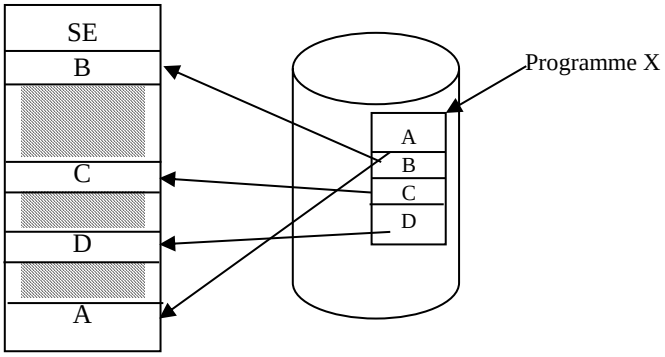
Compactage de MC : pour éviter les fragments inutilisables créés durant l'allocation des processus, un algorithme de compactage permet de rassembler les partitions libres en une seule partition avec une taille importante. Pour effectuer cette opération, il faut que les programmes à déplacer soient relogeables.

2/ Modes utilisant la mémoire virtuelle :

a/ La mémoire virtuelle :

Le principe de la mémoire virtuelle consiste à tendre l'espace physique de la MC à un espace mémoire plus grand, en utilisant une partie de la mémoire secondaire, exactement le disque dur. Les programmes sont dispersés entre la MC et la MS.

Les programmes sont chargés au début de lancement dans la MS, ensuite ils sont découpés en blocs et ils sont ramenés de la MS bloc par bloc vers la MC, le moment d'exécution de chaque bloc.



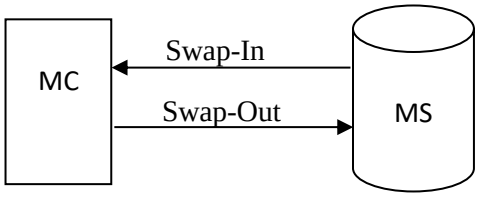
L'utilisation de la mémoire virtuelle a comme objectifs :

- Exécution des programmes lorsqu'ils sont partiellement en MC.
- Le chargement et l'exécution de programmes qui ont une plus grande taille par rapport à la MC.
- Une augmentation du degré de multiprogrammation.

b/ Le swapping (va-et-vient) :

C'est une opération utilisée dans le cas de la mémoire virtuelle, telque :

- Un programme est chargé dans sa totalité dans la MS ensuite il est transféré partie par partie dans la MC (Swap-In)
- Il s'exécute soit jusqu'à sa terminaison, ou son blocage.
- Dans le cas du blocage, le programme peut être transféré en MS (Swap-out)



Pour gérer l'espace de ces deux mémoires ainsi que pour faciliter les opérations de swapping, le système utilise l'une des techniques suivantes :

- La pagination
- La segmentation
- La segmentation paginée

***c/ La pagination* :**

On parle de la pagination, lorsque :

- La mémoire centrale est organisée en blocs de tailles égales et fixes appelés **pages physiques** ou **cadres de pages**.
- Les processus sont divisés en blocs appelés **pages virtuelles** ayant la même taille que les cadres de pages.

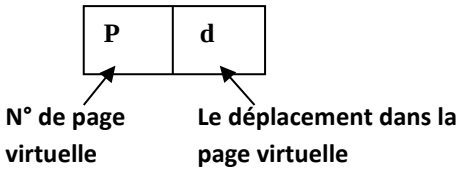
Alors les programmes sont mis dans la mémoire secondaire dans une zone swap dont on manipule un espace d'adressage virtuel. L'exécution d'un programme se fait en ramenant de la MS à la MC, page virtuelle par page virtuelle le moment de sa référence dans un cadre de page libre.

Où le processeur génère l'adresse de l'instruction suivante qui est une adresse virtuelle dont il faut référencer la page contenant cette adresse.

- Correspondance entre l'adresse virtuelle et l'adresse physique :

Lorsque la mémoire virtuelle est utilisée, l'adresse référencée soit logique, elle n'est pas mise directement dans le bus d'adresse, mais elle est envoyée à une unité physique appelée « Unité de gestion de mémoire MMU » qui a comme rôle de faire la correspondance entre cette adresse et son adresse réelle dans la MC, tel que :

- Le processeur génère une adresse virtuelle.
- Le MMU transforme cette adresse à un couple :



- Le gestionnaire de mémoire consulte **la table de pages** du programme en exécution pour savoir l'existence de cette page dans la MC.
- Si cette page n'existe pas dans la MC, il signale **un défaut de page**, et il la charge de la MS vers la MC.
- Il fait la correspondance entre le déplacement virtuelle et l'adresse physique de l'instruction dans la MC,
- Maintenant, cette adresse physique est transférée dans le bus d'adresses.

Table des pages : c'est une table contient toutes les informations nécessaires pour manipuler les pages, elle se trouve dans la MC incluse dans le **Bloc de Contrôle de Processus**, elle contient :

- Le numéro de page.
- L'adresse dans la MS → S.
- Un bit indiquant son existence dans la MC → R.
- L'adresse de cette page si elle est chargée en MC → PR.
- Un bit indiquant si la page est modifiée ou non → M.
- Un bit de protection.
- Un bit de référence.

➢ Gestion de la mémoire virtuelle dans la pagination :

1/La recherche : le gestionnaire cherche l'existence de la page virtuelle pour la charger en MC soit dans la demande du processeur, soit par anticipation (prédiction d'utilisation).

2/Le placement : à quel emplacement la page doit être placée en MC.

3/Le déplacement : quelle page doit quitter la MC, si elle est saturée, pour charger une nouvelle page référencée, le gestionnaire cherche une page victime à remplacer selon une des stratégies :

- Stratégie FIFO → la plus ancienne est remplacée.

- Stratégie LRU → Least Recently Used (moins récemment utilisée).
- Algorithme optimal → prédiction de la page qui ne sera pas référencée durant le plus grand temps).
- Stratégie NRU → Not Recently Used dans un temps.
- Seconde Chance → amélioration du FIFO, la première entrée + elle n'est jamais référencée.
  - ➢ Avantages et inconvénients de la pagination :

Avantages :

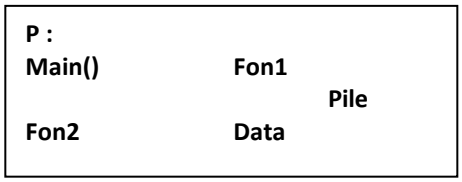
- Simplicité de chargement des pages virtuelles dans des pages physiques.
- Simplicité de correspondance entre l'adresse physique et l'adresse virtuelle.
- Le programmeur ne doit pas connaitre la technique utilisée par le gestionnaire

Inconvénients :

- Augmentation de la taille de la table de page par rapport à la taille des programmes
- Le taux élevé des défauts de pages.
- Les données et les procédures ne sont pas séparées, ce qui complique leurs partages entre différents programmes.
- La protection des données et de procédures n'est pas faite d'une manière simple et séparée.

***d/ La segmentation* :**

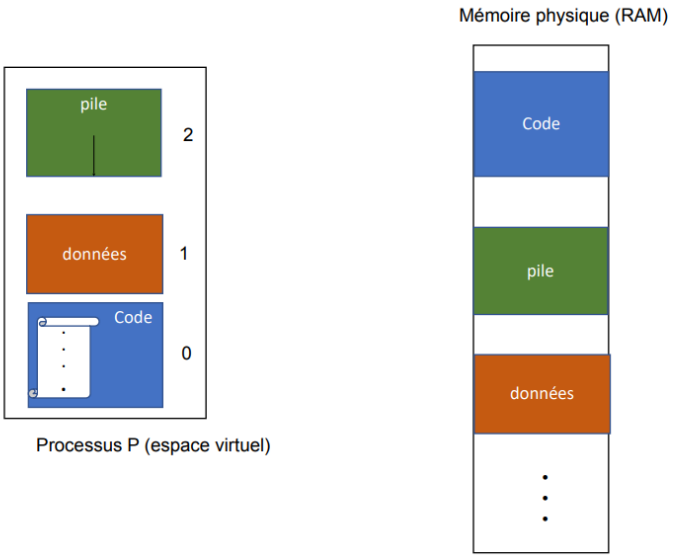
Une stratégie qui reproduit le découpage dans la mémoire tel qu'il est décrit par l'utilisateur et le compilateur (découpage logique)

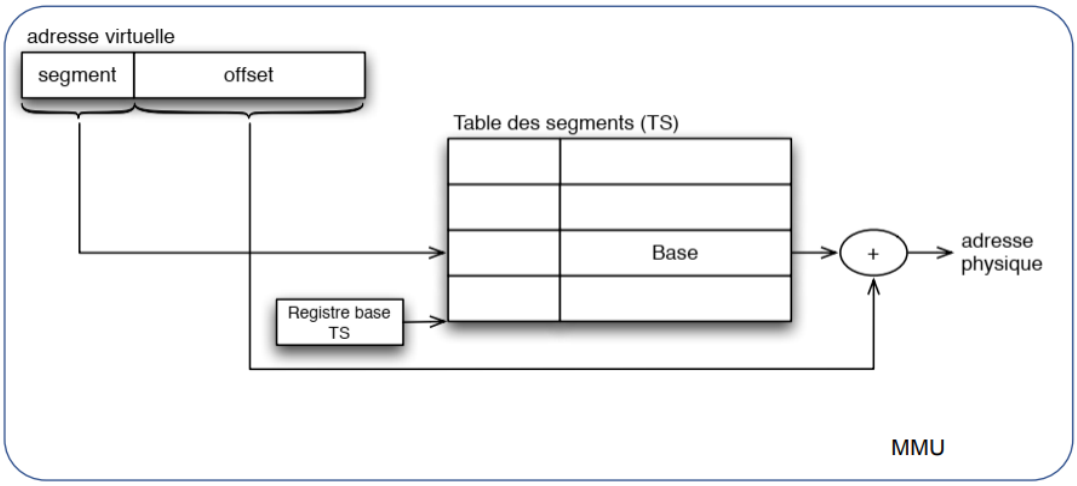


Où :

- La mémoire centrale est organisée en blocs de tailles variables →**Segments physiques.**
- Chaque processus est découpé en unités logiques → Segments virtuelles (les variables globales, la pile d'appel des procédures, chaque procédure ou fonction, les variables locales, le programme principale).

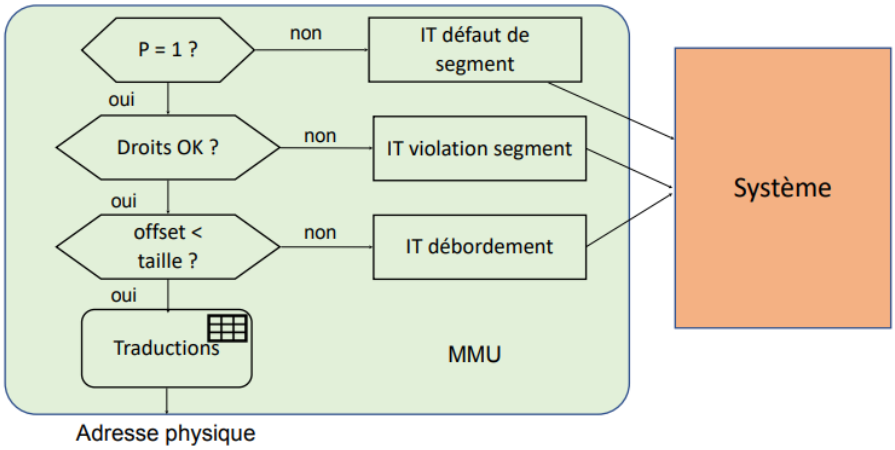
N.B : les segments constituant un programme sont construits lors de la compilation.





Une entrée de la table des segments (P, Droits, Taille, Base)

- P : présence (0 : non présent, 1 : présent) ;
- Droits : r (Read), w (Write), x (eXecutable). • La MMU (processeur) vérifie pour une adresse



Exemple de segmentation dans l'architecture 8086 : L'espace mémoire adressable par le 8086 est de 2^{20} = 1 048 576 octets = 1 Mo (20 bits d'adresses). Cet espace est divisé en **segments**. Un segment est une zone mémoire de 64 Ko (65 536 octets) définie par son adresse de départ qui doit être un multiple de 16.

Dans une telle adresse, les 4 bits de poids faible sont à zéro. On peut donc représenter l'adresse d'un segment avec seulement ses 16 bits de poids fort, les 4 bits de poids faible étant implicitement à 0.

Pour désigner une case mémoire parmi les 2^{16} = 65 536 contenues dans un segment, il suffit d'une valeur sur 16 bits.

Ainsi, une case mémoire est repérée par le 8086 au moyen de deux quantités sur 16 bits :

• l'adresse d'un segment ;

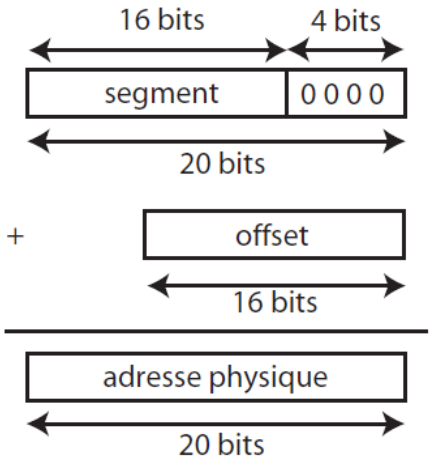
• un déplacement ou **offset** (appelé aussi **adresse effective**) dans ce segment.

Cette méthode de gestion de la mémoire est appelée **segmentation de la mémoire**.

La donnée d'un couple (segment,offset) définit une **adresse logique**, notée sous la forme

segment : offset.

L'adresse d'une case mémoire donnée sous la forme d'une quantité sur 20 bits (5 digits hexa) est appelée **adresse physique** car elle correspond à la valeur envoyée réellement sur le bus d'adresses A0 - A19.

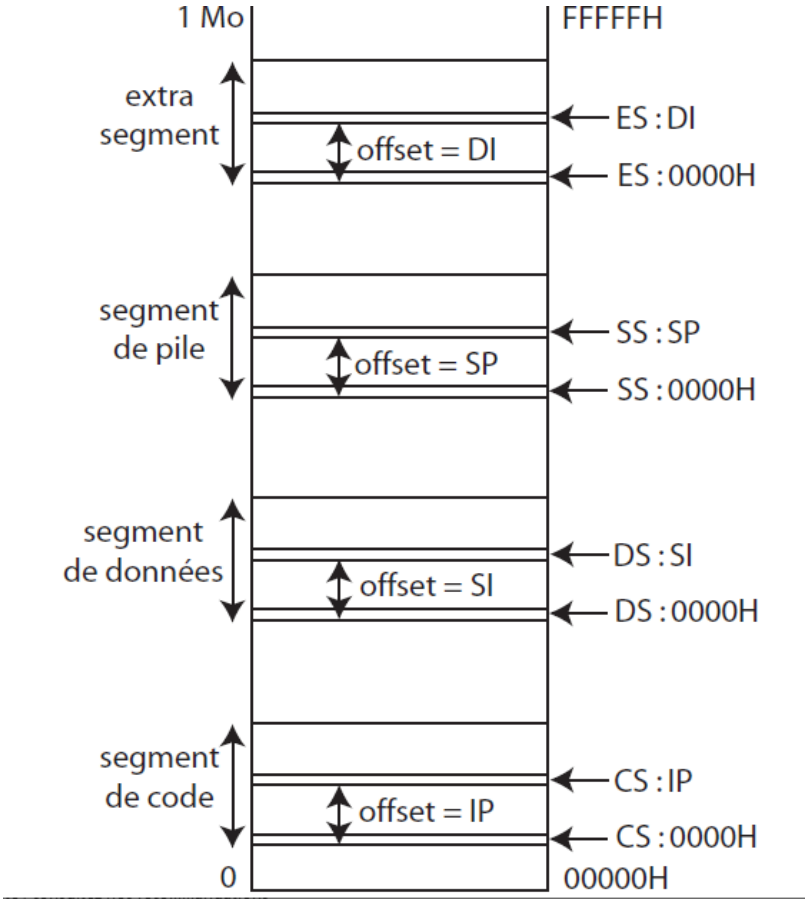


adresse physique = **16** × **segment** + **offset**

A un instant donné, le 8086 a accès à 4 segments dont les adresses se trouvent dans les registres de segment CS, DS, SS et ES. Le segment de code contient les instructions du programme, le segment de données contient les données manipulées par le programme, le segment de pile contient la pile de sauvegarde et le segment supplémentaire peut aussi contenir des données.

Le registre CS est associé au pointeur d'instruction IP, ainsi la prochaine instruction à exécuter se trouve à l'adresse logique CS : IP.

De même, les registres de segments DS et ES peuvent être associés à un registre d'index. Exemple : DS : SI, ES : DI. Le registre de segment de pile peut être associé aux registres de pointeurs : SS : SP ou SS : BP.



***e/ <u>La segmentation paginée</u>* :**

Cette technique a comme objectif de regrouper les avantages des deux techniques précédentes, tel que :

- Le programme est découpé, en premier lieu, en segments logiques avec des tailles variables, chaque segment est découpé en pages virtuelles de la même taille.
- La mémoire centrale est découpée en pages de la même taille.
- L'adresse virtuelle est un triple constitué du numéro de segment, numéro de page et le déplacement dans la page.

V/ <u>La protection mémoire</u> :

Tout système doit offrir un mécanisme de protection contre les accès non autorisés à des blocs de mémoire, suite à des erreurs de programmation ou autre. Les processus doivent être protégés entre eux contre leurs activités. Deux modes sont proposés :

1/ Protection du tout ou rien :

L'accès à une zone mémoire pour un processus donné est soit complètement autorisé, soit interdit, c'est-à-dire en sachant les limites début et fin de cette zone, on indique si ce qui existe entre ces limites soit protégé ou non.

2/ Protection spécifiée :

Permet de spécifier les opérations autorisées sur une zone mémoire, les opérations concernées sont :

- La lecture (Read).
- L'écriture (Write).
- L'exécution (Execute).

Où, on spécifie trois bits de protection RWE avec des valeurs différentes selon le niveau de protection (Exemple : 000 → accès interdit, 001 → exécution seule, …)