GESTION DE LA MEMOIRE

MEMOIRE CENTRALE (MC)

MEMOIRE SECONDAIRE (MS)

1. Hiérarchie et différents types de mémoire

On distingue 3 niveaux de mémoire: la mémoire de travail appelée encore mémoire centrale (volatile), la mémoire secondaire (disque et disquette) et la mémoire de masse (bande et cartouche).

La mémoire secondaire et de la mémoire de masse constituent ce qu'on appelle la **mémoire permanente**. Celle-ci est utilisée pour le stockage des informations sous forme de fichiers (données, programmes sources, exécutables, etc). La mémoire de masse contient l'information peu utilisée de la mémoire permanente (archivages, sauvegardes, données, etc).

La **mémoire centrale** contient des programmes (instructions + données). Le chargement en mémoire centrale d'un fichier exécutable (processus) donne une image en mémoire centrale différente de celle en mémoire secondaire (stockage). L'Unité Centrale utilise des **registres** en entrée et en sortie de ses unités fonctionnelles (addition, multiplication, etc). L'information est transférée de la mémoire centrale vers les registres et réciproquement.

Le concept de mémoire centrale pose un certain nombre de problèmes: limitation en performance (l'unité de traitement est plus rapide que les transferts mémoire), conflits d'accès à la mémoire centrale, limitations de l'espace physique de l'adressage, etc. Différentes techniques permettant d'accélérer le traitement des instructions (lecture anticipée des instructions et des opérandes par exemple) ont été ainsi développées:

- La **mémoire multiport** possède plusieurs dispositifs d'adressage et d'accès indépendants. Chaque port dispose d'un bus d'adresse, de signaux de commande et de buffers de données propres. Exemple: la présence d'1 port d'écriture et de 2 ports de lecture peut accélérer fortement le traitement d'une opération portant sur 2 opérandes.

- Avec la **mémoire entrelacée**, 2 adresses successives sont situées dans 2 blocs physiques différents (en général consécutifs). Ces blocs, appelés aussi bancs mémoires, fonctionnent en parallèle. Une gestion des conflits d'accès au même banc est à effectuer.

- Les **mémoires associatives** sont utilisés pour rendre, en un seul cycle machine, la valeur associée à une clé d'accès.

- La **mémoire cache** est une mémoire plus rapide que la mémoire centrale. Elle contient une copie d'une zone de mémoire centrale et permet de diminuer les temps d'accès en accélérant le traitement des instructions par l'unité de commande.

- La **mémoire virtuelle** repousse les limites de l'adressage réel de la mémoire centrale. La mémoire centrale déborde en fait sur la mémoire secondaire et le rôle de la mémoire centrale devient pratiquement celui d'un cache de la mémoire secondaire. **Pagination** et **Segmentation** sont 2 techniques de gestion de la mémoire virtuelle. La Pagination effectue le découpage physique en pages de la mémoire virtuelle. La Segmentation permet un découpage logique des programmes en segments logiques pouvant être chargés en mémoire centrale plus ou moins indépendamment. On distingue les segments de codes (instructions) et les segments de données (variables) associés à un segment de codes ou partagés.

Dans une architecture multiprocesseurs, on parle de mémoire distribuée et de mémoire partagée. La mémoire partagée ...

L'occupation de la mémoire centrale par plusieurs processus implique un partage et un découpage en **zones**. Il en résulte des stratégies d'**allocation** et de **libération** de ces zones, ainsi que de lecture et écriture d'un mot dans une zone. Selon que l'information

relative à une même entité se trouve dans une zone contiguë ou non, le **modèle d'allocation** mémoire est dit **contiguë** ou **non contiguë**. Différentes stratégies et mécanismes d'accès sont associés à chaque modèle.

Dans ce qui suit, un **bloc** représente le plus petit nombre de mots contigus que l'on peut allouer. Lorsqu'une entité n'occupe pas exactement un nombre entier de blocs, on parle de **fragmentation interne**: il reste des mots libres non utilisés à la fin du dernier bloc. La libération d'un ensemble de blocs (d'une zone) provoque une fragmentation appelée **fragmentation externe**.

2. Mémoire cache

La plupart des machines ont adopté une organisation à hiérarchie mémoire à au moins 2 niveaux: le niveau supérieur et le niveau inférieur. le niveau inférieur est la mémoire centrale. Le niveau supérieur est près du processeur. Le mot **Cache** est le nom utilisé à l'origine pour désigner le(s) niveau(x) de mémoire entre l'unité centrale et la mémoire centrale (exemple cache interne et externe, de niveau 1 ou 2 sur les Pentiums).

La **mémoire cache** est une mémoire beaucoup plus rapide que la mémoire centrale. Elle contient une copie d'une zone de mémoire centrale et permet de diminuer les temps d'accès en accélérant le traitement des instructions par l'unité de commande. L'échange d'information entre les deux niveaux s'effectue par bloc. Seulement une partie des blocs du niveau inférieur sont présents dans le niveau supérieur. On appelle *succès* un accès à un bloc *présent* dans le cache supérieur. La non présence d'un bloc est appelé *défaut* de cache.

On distingue en général le cache d'instructions et le cache de données. Un cache d'instructions doit pouvoir contenir l'ensemble des instructions consécutives d'une boucle de programmation. Il peut être relativement petit. Un cache de données doit pouvoir contenir des zones de données différentes de la mémoire centrale (les tableaux de données). Il est relativement grand. Nous ne nous intéressons ici qu'à ce dernier type de cache.

Lors d'une requête à une donnée (lecture), la présence de cette donnée dans le cache est testée. Si la donnée est présente (*succès*), elle est lue à partir du cache. Si elle est absente (*défaut*), la zone de la mémoire centrale à laquelle elle appartient est chargée dans le cache (dans une zone libre ou en remplacement d'une zone non utilisée).

La gestion des données présentes est réalisée à l'aide d'une table indiquant les zones de mémoire centrale (appelées en général **pages**) dupliquées dans le cache. Chaque bloc du cache possède une étiquette représentant le numéro réel du bloc de la mémoire principale. Pour tester si une donnée d'une adresse mémoire centrale donnée est présente dans le cache ou non, il suffit de consulter cette table. Différentes techniques sont utilisées : mémoires associatives, hash coding, etc. L'intérêt des mémoires associatives est de permettre une recherche d'un numéro de bloc en parallèle (et non en série).

Il y a au moins 3 technologies de cache différentes :

- cache à correspondance directe: chaque bloc a une seule place possible (modulo),
- cache totalement associatif: un bloc peut être placé n'importe où
- cache par ensemble associatif (mixte): un bloc peut être mis dans un ensemble restreint de places (affectation à un ensemble, puis n'importe où).

Lorsqu'un bloc est à mettre dans le cache, il vient en général remplacer un bloc déjà existant qu'il faut choisir. Les stratégies les plus courantes pour le remplacement sont : le hasard, le choix du plus ancien (FIFO) ou le choix du moins utilisé (**L**east **R**ecent **U**sed).

La table du cache est aussi utilisée pour gérer les blocs de mémoire centrale à remettre à jour après écriture d'une donnée (dans le cache). En effet la modification d'une donnée dupliquée dans le cache implique une mise à jour de la vraie valeur en mémoire centrale.

Si l'écriture d'une donnée est moins fréquente en général que la lecture, elle peut entraîner une séquence pénalisante de lecture/modification/écriture du bloc de données de la mémoire principale: défaut de cache (transfert mémoire centrale vers cache), écriture dans le cache, mise à jour de la page de mémoire centrale. Ainsi l'écriture dans un bloc non présent dans le cache peut être réglée de deux manières :

- avec l'écriture attribuée (lecture sur écriture) le bloc est lu dans le cache,
- avec l'écriture non attribuée le bloc est mis à jour directement dans la mémoire principale sans passer dans le cache.

Dans le cas de mise à jour, les techniques couramment utilisées sont : mise à jour simultanée en mémoire principale ou mise à jour différée au moment du remplacement du bloc (quand le bloc est retiré du cache).

Avec le **Write Through**: la mise à jour est effectuée au plus tôt. A chaque modification de la copie, l'original est mis à jour de manière asynchrone (pas d'attente synchronisée). Si les écritures ne sont pas trop rapprochées alors il n'y a pas d'attente de la mise à jour des originaux.

Avec le **Write Back**: la mise à jour est effectuée au plus tard. La bloc du cache contenant la donnée modifiée est invalidée. Lors d'un défaut de cache, la libération de ce bloc pour accueillir le bloc de donnée demandé provoque la mise à jour du bloc original. En général avec cette technique, la taille du bloc est de l'ordre de 16 à 256 mots mémoire. Plus difficile à réaliser que le Write Through, le Write Back donne de meilleures performances pour les monoprocesseurs.

Pour les machines SMP, la technique de Write Back est la plus adaptée, la technique de Back Through générant beaucoup de trafic de données.

Comme on peut le constater, le fonctionnement d'un cache est complexe. Son rôle est vital pour les performances des environnements à multiprocesseurs. Si dans le cas d'un monoprocesseur, un défaut de cache signifie une augmentation des accès de la mémoire centrale, pour un multiprocesseur il peut provoquer un d'étranglement des accès à la mémoire centrale avec une forte baisse des performances.

Un certain nombre d'articles font mention des problèmes de performances dus aux techniques mises en jeu par les caches :

- hiérarchie de caches (nombre de niveaux),
- taille des blocs du cache,
- gestion des pages du cache (direct, associatif),
- stratégies de remplacement des blocs(FIFO, LRU, random)
- stratégies de mise à jour(Write Back, Write Through),
- entrelacement des bancs de mémoire (interleaving),

- modèle de consistance des caches (fort, faible),
- cohérence des caches.

Nous avons décrit la plupart des techniques ci avant. L'entrelacement des bancs de mémoire n'est pas une technique de cache mais une technique (connue) d'accès en parallèle à la mémoire qui peut poser des problèmes dans la synchronisation des mises à jour du cache. Le modèle de consistance n'intervient que dans les environnement multiprocesseurs.

La **consistance** des caches est dite forte (sequential) ou faible (weak) selon que le fait que les processeurs ont (ou n'ont pas) à un moment donné la même image de la mémoire (partagée ou distribuée). Le modèle de consistance forte est équivalent à un déroulement séquentiel des opérations. Le modèle de consistance faible n'est en fait consistent qu'à certains moments donnés.

Il y a en fait deux techniques adoptées, lorsqu'un processeur écrit des données dans son cache:

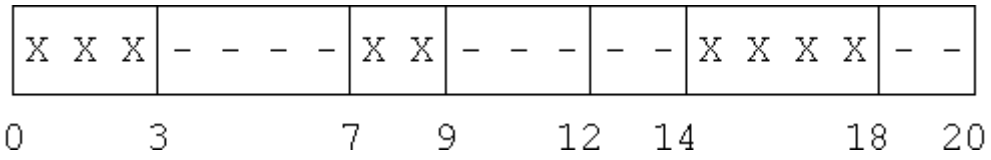
- invalider les copies de ces données dans les caches des autres processeurs,
- mettre à jour les copies dans les caches de tous les autres processeurs.

Les problèmes à résoudre sont en fait :

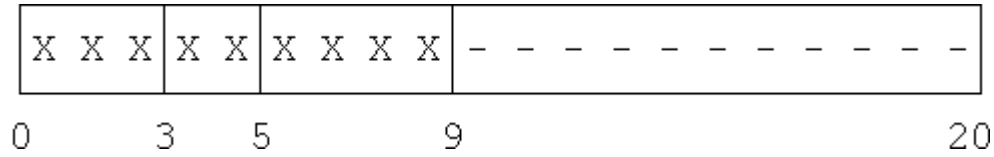
- *stale value in local cache*
- *stale value in main memory*

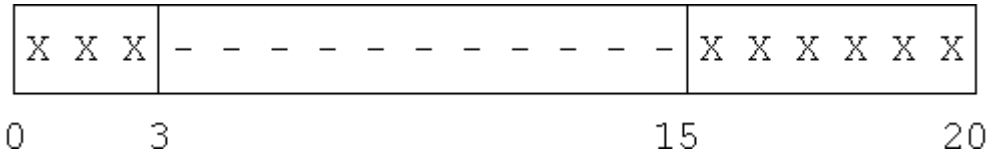
3. Modèle d'allocation contiguë (MC ou MS)

Une même entité (processus en mémoire centrale ou fichier en mémoire secondaire) occupe des blocs contigus. Exemple avec 3 entités:



Différentes techniques de récupération de place (**compactage**) existent. Le coût du compactage dépend du nombre de blocs déplacés. Exemples:



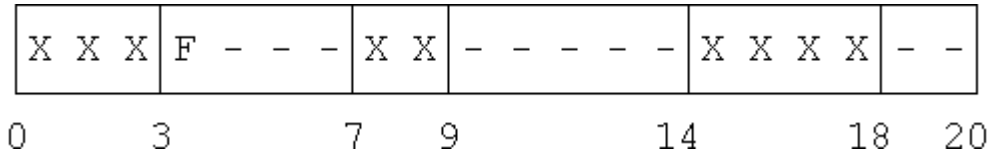


L'opération de compactage et la fréquence des compactages ralentit le fonctionnement du système. C'est pourquoi on s'intéresse aux **stratégies de placement** permettant d'éviter des compactages systématiques. Le choix d'une zone libre à allouer doit s'effectuer rapidement, minimiser la fragmentation externe et garantir une utilisation globale optimisée. Les stratégies de placement suivantes permettent d'allouer une zone d'un bloc

3.1. STRATÉGIE DE 1ÈRE ZONE LIBRE (FIRST FIT)

La liste des zones libres est parcourue et la première zone libre de taille suffisante (égale ou supérieure) est sélectionnée. Cette stratégie est rapide

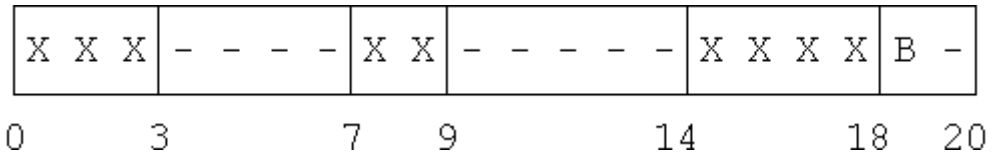
F = FIRST FIT



3.2. MEILLEUR AJUSTEMENT (BEST FIT)

La liste des zones libres est parcourue et la zone libre provoquant une fragmentation externe minimale est sélectionnée. Cette stratégie plus lente que la précédente apporte un confort d'utilisation globale mais provoque un grand nombre de petits fragments.

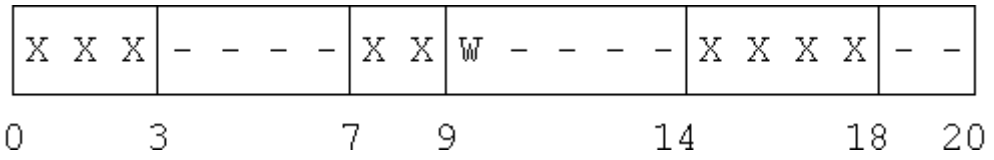
B= BEST



3.3. PLUS GRAND RÉSIDU (WORST FIT)

La zone la plus grande est choisie. Les fragments externes produits sont grands et peuvent être utilisés ultérieurement.

W = WORST FIT



4. Modèle d'allocation contiguë en Mémoire Centrale

4.1. Liaison des adresses

Lorsqu'un fichier exécutable est produit, les références aux procédures, aux variables locales ou globales sont codées logiquement. L'installation d'un programme en mémoire centrale implique la mise à jour des adresses dans les programmes. Certaines références sont partiellement satisfaites à l'édition de liens. C'est au chargement du programme en mémoire centrale que les adresses "Mémoire Centrale" peuvent être connues. L'association d'un objet et de sa représentation physique s'appelle **liaison**.

L'adressage utilisé dans le fichier exécutable peut être relatif ou absolu. L'**adressage relatif** est utilisé pour permettre à un programme de se loger facilement à n'importe quel emplacement (adresse) en MC. L'adressage relatif peut consister soit au codage d'une adresse relative par rapport au début du programme ou bien d'une adresse relative par rapport à l'instruction en cours (déplacement). L'adressage absolu ne permet pas à un programme d'être relogé aisément. Le programme est dit non relogeable.

Exemple fichier relogeable (COM) et absolu (EXE MZ)

4.2. Systèmes monoprogrammés en mémoire réelle

Dans le cas de l'adressage absolu, chaque adresse absolue est comparée à une adresse contenue dans registre spécial : le Registre Barrière permettant la protection de la zone où se trouve le système d'exploitation.

Dans le cas de l'adressage relatif, il faut ajouter à l'adresse relative l'adresse de début du programme contenue dans une registre spécial : le Registre de Base (dit encore de Garde).

4.3. Technique de recouvrement (overlays)

Le chargement et le retrait des segments de programmes est effectué dynamiquement. Le découpage en segment est à la charge du programmeur. En général, l'adressage est effectué relativement par rapport à chaque de segment (dont l'adresse est connue dynamiquement). Le programme principal, chargé en premier, reste en général en mémoire car il contient des variables globales utilisées par les autres segments. Le système peut être mono ou multiprogrammé. La technique de recouvrement peut être aussi utilisée pour l'allocation non contiguë (pagination ou segmentation.

4.4. Systèmes multiprogrammés

Les **Echanges superposés** s'effectuent à l'aide de 2 zones tampons. Le chargement du programme devant s'exécuter après le programme courant est effectuée dans une zone tampon n°1 en parallèle avec l'exécution du programme courant. La commutation provoque le transfert du programme courant dans une zone tampon n°2 et le transfert de la zone tampon n°1 en lieu et place de la zone libérée.

La gestion de la mémoire centrale en **Partitions Fixes** pose des problèmes de fragmentation et de succession en MC. Avec des **Partitions variables**, des problèmes dus à la variation dynamique surgissent.

5. Modèle d'allocation contiguë en Mémoire Secondaire

Exemple: Extension d'une zone ou mise à jour dans le cas d'un fichier ;

6. Modèle d'allocation non contiguë (MC ou MS)

Dans ce modèle, les différents fragments sont répartis dans des zones non consécutives. Des techniques de chaînage ou d'indexation sont utilisées pour parcourir ou retrouver un fragment appelé bloc.

Dans la technique de **chaînage** seul la référence au premier bloc est connu. A la fin de chaque bloc se trouve la référence au bloc suivant. Une technique de double chaînage (avant et arrière) est en général utilisée pour éviter les pertes d'information et le positionnement en arrière (backspace) au niveau des enregistrements logiques (records) pour les fichiers. En mémoire centrale, le chaînage s'effectue le plus souvent sur les références et non sur les blocs eux mêmes.

La technique de l'**indexation** est plus rapide et permet d'éviter le chaînage dans les blocs. Le choix de la taille de la table d'index est délicat: un compromis est à trouver entre taille maximale et taille moyenne.

Ces 2 techniques peuvent être utilisées conjointement, exemple: liste chaînée de bloc d'index.

7. Modèle non contiguë en Mémoire Centrale

7.1. SEGMENTATION

Les segments sont le reflet de la vision de l'utilisateur. La longueur des segments est variable et dépend du découpage effectué. Contrairement à la technique de recouvrement les segments sont typés: on distingue les segments de codes (programmes, bibliothèques, etc) et les segments de données. La longueur des segments est variable et dépend de l'utilisation.

Mémoire virtuelle, Segment Réentrant.

7.2. PAGINATION

Pages (taille fixe), cadres, partage et protection des pages

Localité dans le temps : réutilisation

Localité dans l'espace : utilisation des voisins

Pb. du partage et de la protection des pages

8. Modèle d'allocation non contiguë en Mémoire Secondaire

Non traité.

9. Gestion de la mémoire centrale sous MS-DOS

C'est en 1981 qu'est commercialisé l'IBM PC utilisant le microprocesseur 8088 d'Intel, version dérivée du 8086 avec un bus de données sur 8 bits. Ce processeur ne pouvant adresser que 1024 Ko, il fut décidé dans un premier temps de réserver 512 Ko pour le système (BIOS) et 512 Ko pour l'utilisateur (programmes applicatifs et programmes résidents).

La partie utilisateur fut par la suite enrichie d'un bloc de 128 Ko donnant naissance à la fameuse limitation de 640 Ko. Ce n'est qu'à partir du 80286 que l'adressage atteignit les 16 Mo. Avec les 386 et 486, il est passé à 4 Go.

Il y a essentiellement 2 types d'applications s'exécutant sur PC: les applications DOS et les applications WINDOWS. Les dernières utilisent toute la mémoire centrale disponible, alors que les applications DOS restent tributaires de la barrière des 640 Ko.

La **Mémoire Conventionnelle** est la zone de 0 à 640 Ko permettant le chargement des programmes s'exécutant en mode réel 16 bits.

La **Mémoire Supérieure** (Upper Memory Area) est la zone de 640 à 1024 Ko (TOP384), c'est une zone partagée avec la ROM (mémoire vidéo, carte réseau, etc).

Notons qu'à partir du 286 et à la suite d'un erreur de conception, une zone de 64 Ko située après le premier Méga-octet est directement adressable: celle-ci est appelée HMA = High Memory Area.

La **Mémoire étendue** est le mémoire après le premier Méga-octet. Elle est utilisée en mode protégé (exécution du code chargé en mémoire étendue) et nécessite un gestionnaire comme par exemple HIMEM.SYS.

Afin d'utiliser la mémoire supérieure au premier Méga-octet, l'association Lotus, Intel et Microsoft a établi en 1984 la norme LIM-EMS permettant d'utiliser de la **mémoire paginée** (EMS = Expanded Memory Specification): les pages EMS sont lues et écrite par l'intermédiaire d'un emplacement spécifique en mémoire haute (Upper Memory Blocks). La technique de mémoire paginée EMS est utilisée pour le stockage de donnée en mémoire étendue par segment de 64 Ko.

La norme XMS définit un adressage de la mémoire étendue. On distingue : les zones EMB (Extended Memory Blocks), les UMB (Upper Memory Blocks) et la mémoire haute HMA (High Memory Area).

La norme DPMI définit une technique permettant à des programmes s'exécutant en mode protégé de cohabiter avec un gestionnaire de mémoire étendue.

Dans la pratique, les processeurs fonctionnent selon différents modes: le **mode réel** est un mode compatible avec les 8088 (PC-XT d'origine) et subit les contraintes inhérentes à ce processeur, le **mode protégé** permet d'adresser toute la mémoire appelée mémoire étendue (XMS = eXtended Memory Specification), le **mode virtuel** (spécifique au 386/486) permet aux applications DOS de travailler en mode protégé dans

l'environnement du mode réel. C'est à l'aide de ce dernier mode que la gestion de mémoire paginée EMS est émulée dans la mémoire étendue XMS.

Windows utilise la mémoire étendue et s'exécute en mode protégé.

La mémoire virtuelle sous Windows 3.1 (Panneau Configuration, icône 386 étendu, option mémoire virtuelle) possède 3 modes de fonctionnement. La quantité de mémoire virtuelle ne peut dépasser 4 fois la quantité de mémoire vive. Le mode permanent réserve un espace disque Permanent constitué de secteurs contigus. Le mode Temporaire alloue dynamiquement le fichier d'échange. Le mode "Aucun fichier d'échange" peut être utilisé à partir de 16 Mo de mémoire vive. L'inhibition de la mémoire virtuelle permet d'éliminer certains conflits.

La mémoire cache primaire (= registres) est gérée par le contrôleur de mémoire MMU (Memory Management Unit).

Pour information, le DX (1985) fonctionne en interne sur 32 bits (4Go), processeur de 275.000 transistors mais le MMU et le coprocesseur sont séparés. Fréquence d'origine 16MHz.