Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes d'exploitation

La Gestion de la Mémoire

Jacques Supcik | 2018/2019 | T-2adfg

l05-handouts | 2017-10-12

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Le rêve des programmeurs

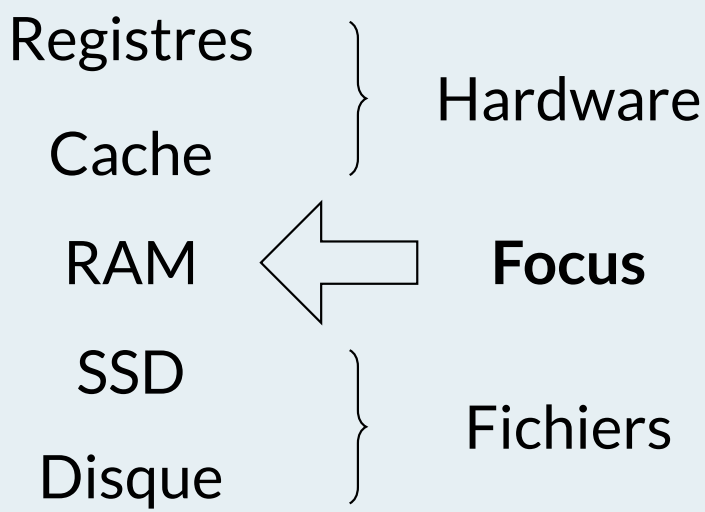
- Disposer d'une mémoire privée
- Infiniment grande
- Infiniment rapide
- Non-volatile (persistante)
- Bon marché

Mais aujourd'hui, ça reste un rêve !

Gestion de la mémoire

Le gestionnaire de mémoire est responsable des deux opérations suivantes :

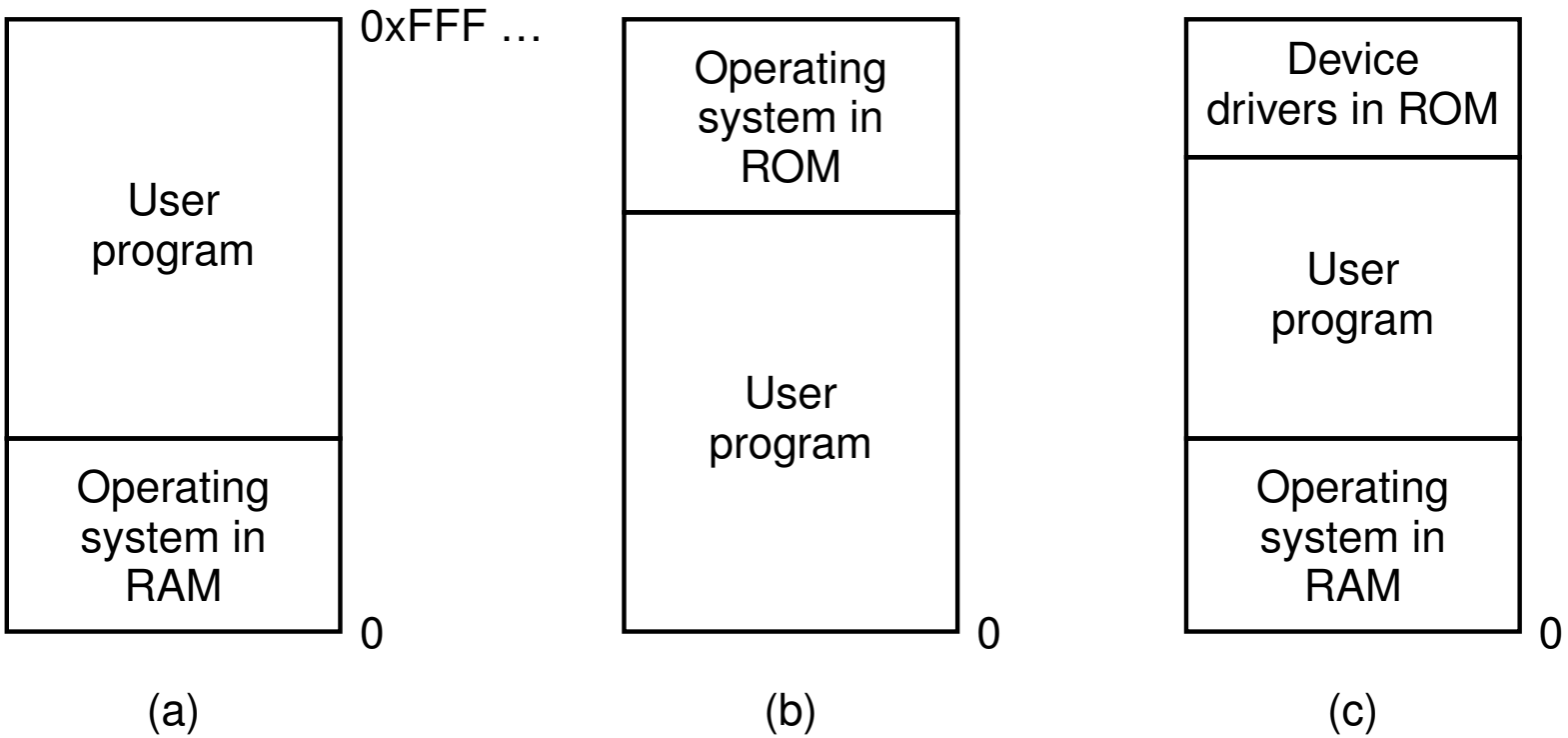
- Allocation de la mémoire
- Dé-allocation de la mémoire



Aucune abstraction de la mémoire

- Utilisation directe de la mémoire
- Un seul programme en mémoire à la fois
- Threads possibles
- On peut avoir plusieurs programmes concurrents si on a recours au «swapping» (échange entre la mémoire et le disque)

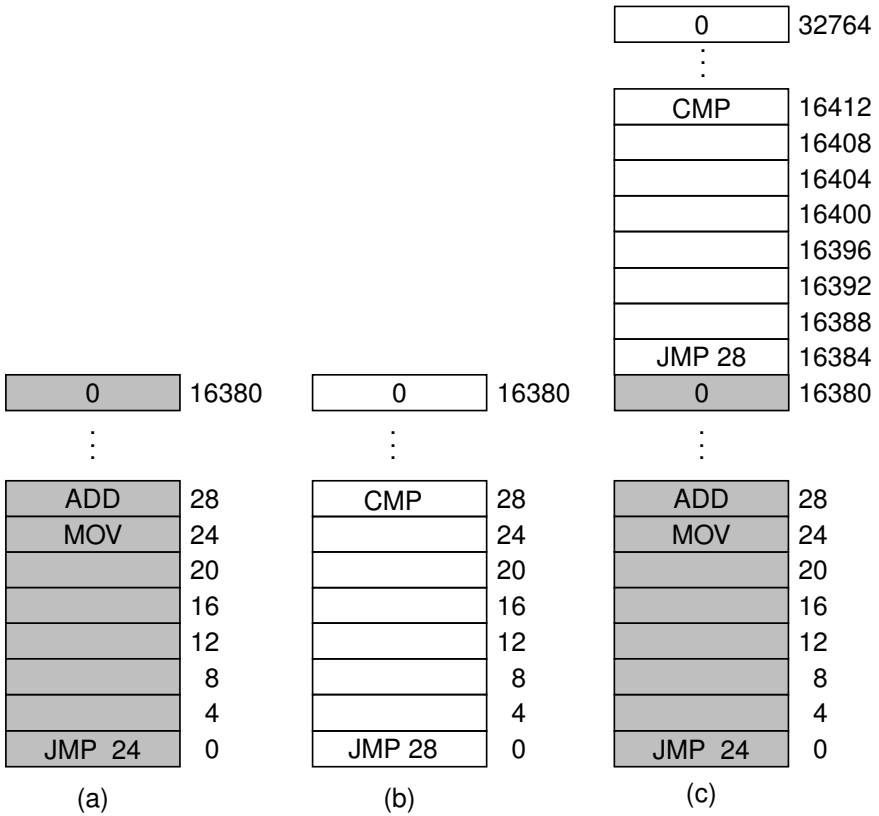
Aucune abstraction de la mémoire



Trois possibilités simples d'organiser la mémoire avec in système d'exploitation et un seul processus utilisateur

Plusieurs programmes sans abstraction

Le problème de ré-allocation



(a) et (b) sont des programmes de 16 kiB. En (c), les deux programmes sont chargés en mémoire.

Plusieurs programmes sans abstraction

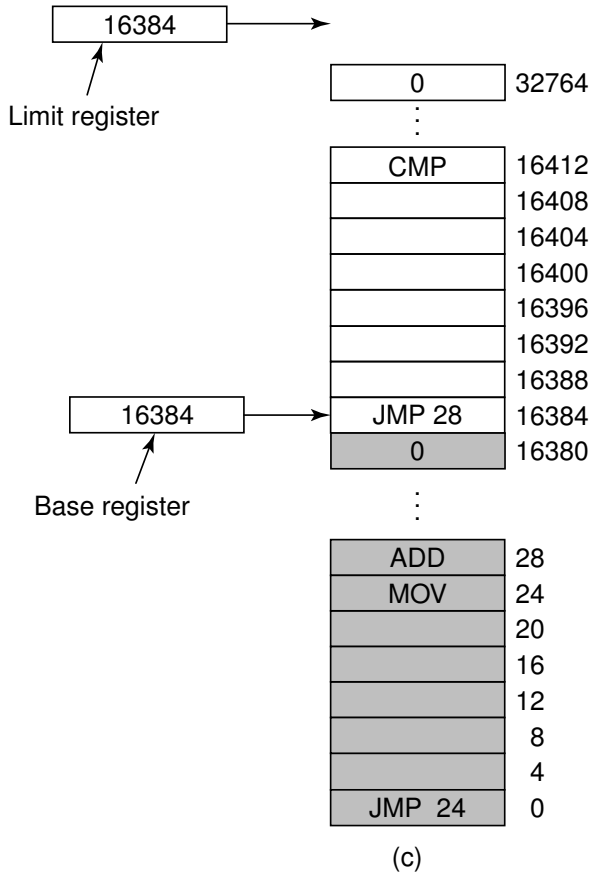
Des solutions au problème de ré-allocation

- Le programme peut être corrigé (patched) par le logeur (linker)
- Dangereux, car un programme peut avoir accès à la mémoire d'un autre programme
- Ce modèle est encore utilisé aujourd'hui dans des systèmes embarqués simples

L'espace d'adressage

- L'espace d'adressage est une abstraction de la mémoire physique
- C'est l'ensemble des adresses qu'un processus peut utiliser
- Chaque programme a son propre espace d'adressage, indépendant de celui des autres
- L'espace d'adressage n'est pas un concept limité à la mémoire (numéros de téléphone, adresses IP, adresses MAC, noms de domaines, …)

Registres de base et limite



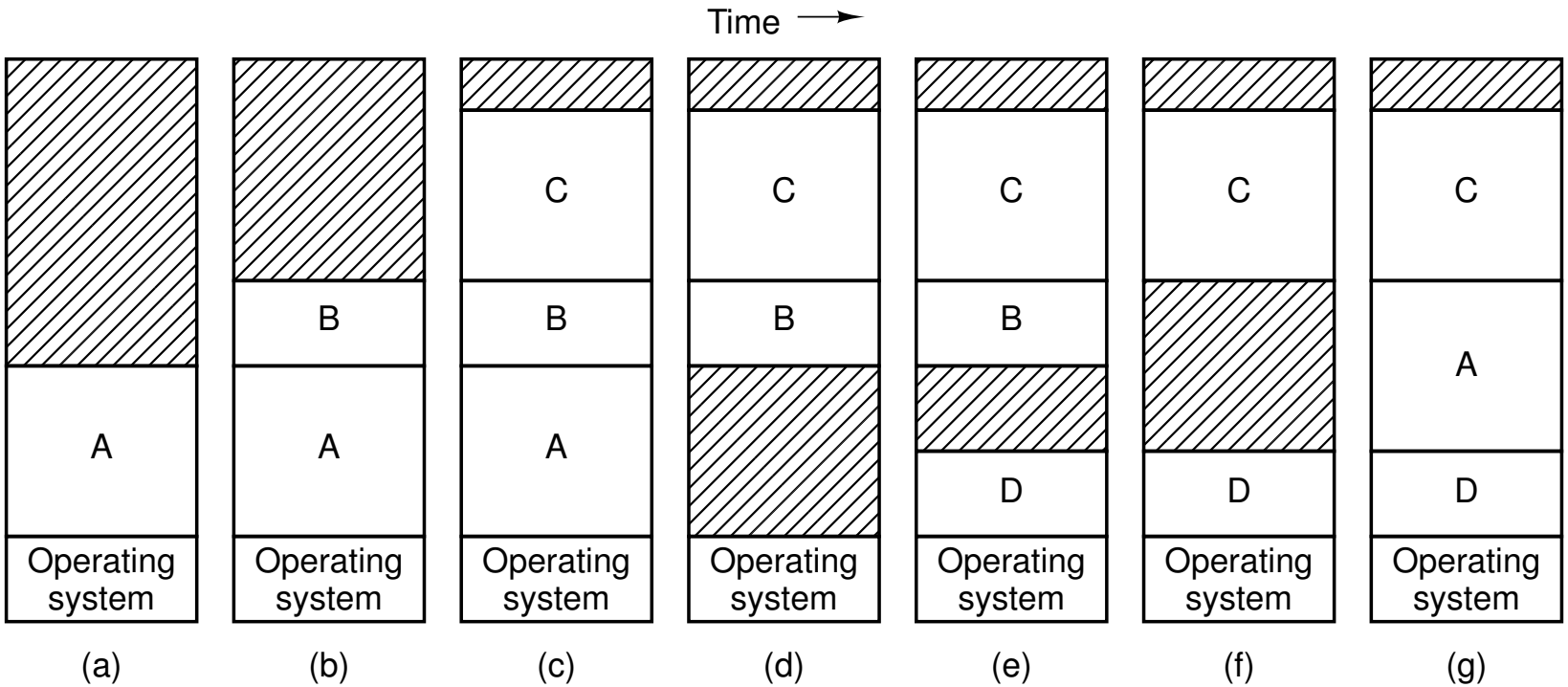
Registres de base et limite utilisés pour attribuer à chaque processus un espace d'adressage.

Registres de base et limite

- Utilisation de registres (hardware) dédiés
- Système assez lent (addition d'adresses et vérifications pour chaque accès mémoire)
- Méthode utilisée par les premiers PC d'IBM (Intel 8088). Le registre de base était le registre BP et il n'y avait pas de registre limite.
- Pour des raisons de sécurité, ces registres ne devraient être modifiables que par le système d'exploitation

Le va-et-vient (swapping)

Solution pour le problème des programmes plus gros que la taille de la mémoire



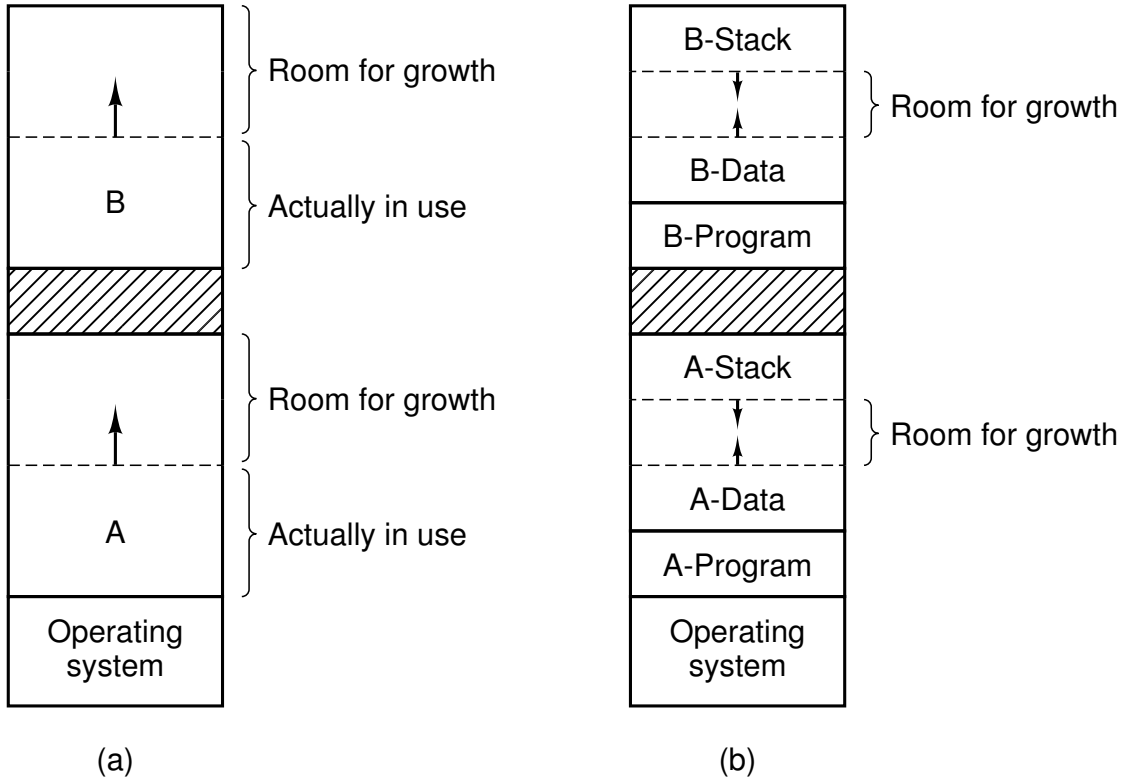
L'allocation mémoire change au gré des processus qui viennent en mémoire et qui la quittent. Les zones grisées indiquent que la mémoire est inutilisée.

Le va-et-vient (swapping)

- Premier problème : La mémoire devient fragmentée
- Solution : le compactage de la mémoire
- Mais cette solution est gourmande en ressources (pour compacter 1 GiB, il faut compter environ 5 secondes)
- Deuxième problème, la taille de la mémoire utilisée par les processus n'est pas fixe (dynamic data segment)
- Solution : allouer plus de mémoire que nécessaire

Le va-et-vient (swapping)

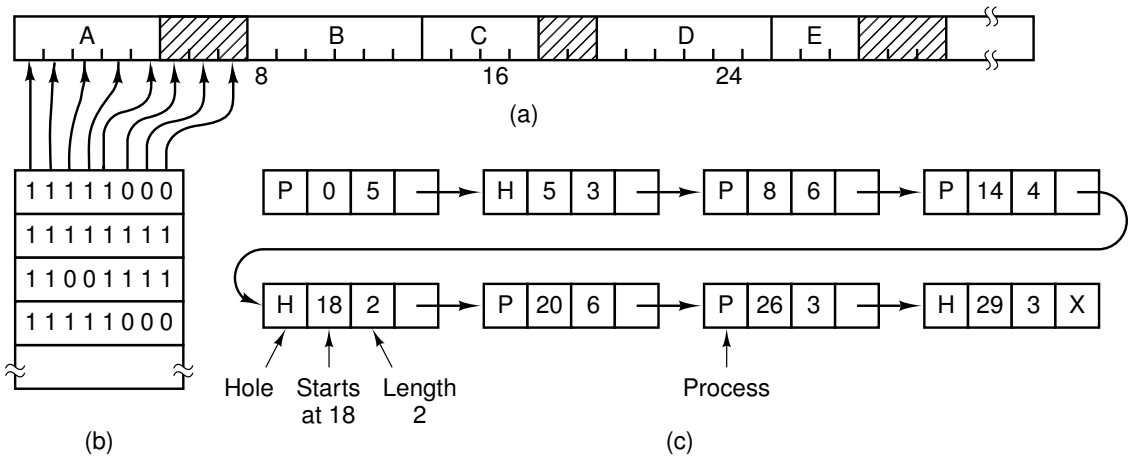
Allocation supplémentaire



- (a) Allocation d'espace pour l'accroissement d'un segment de données.
- (b) Allocation d'espace pour l'accroissement de la pile et d'un segment de données.

Gestion de la mémoire libre

avec une table de bits (Bitmaps)



- (a) Une partie de la mémoire avec 5 processus et 3 trous. Les petites marques verticales indiquent les unités d'allocation. Les zones grisées (valeur 0 dans le tableau des bits) sont des zones libres.
- (b) Le tableau de bits correspondant.
- (c) Les mêmes informations sous forme d'une liste chaînée.

Gestion de la mémoire libre

avec une table de bits (Bitmaps)

- La taille de la table des bits est inversement proportionnelle à la taille des segments (qui va de quelques Bytes à quelques kiBytes)
- Avec des petits segments, la table est grande, mais on a peu de gaspillage dans les blocs
- Avec des grands segments, la table est petite, mais on a beaucoup de gaspillage dans les blocs
- Avec des segments de 4 Byte, 1 bit représente 32 bit. La table des bits prendra $\frac{1}{33}$ de la mémoire totale, soit environ 3%
- Problème : la recherche d'un bloc libre n'est pas efficace (il faut chercher une séquence de «0» dans la table)

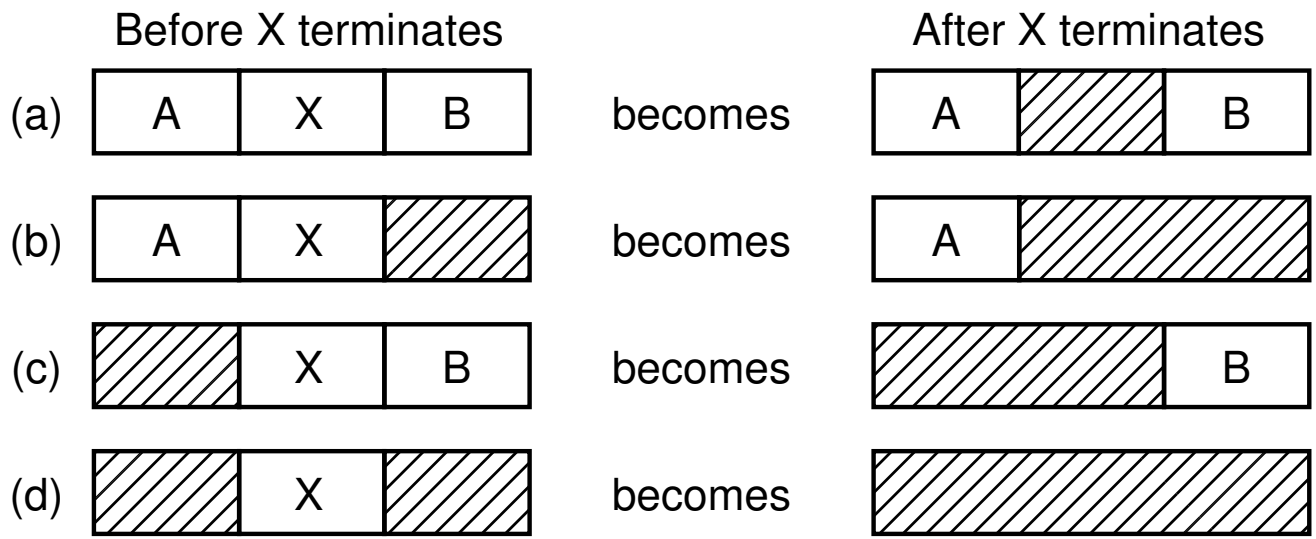
Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)

- Chaque nœud contient un bit qui indique si le bloc correspond à un processus ou à un «trou», l'adresse de début du bloc, ainsi que la taille du bloc.
- Habituellement, l'entrée dans la table des processus contient un pointeur vers le début de la liste
- On peut simplifier la gestion de la liste en utilisant un double chaînage (mais on utilise plus de mémoire)

Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)



4 combinaisons de voisinage pour un processus `X` qui se termine.

Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)

On peut trier la liste par adresse. Dans ce cas, on peut implémenter les algorithmes suivants :

- «First fit» : on recherche, depuis le début de la liste, le premier bloc qui est plus grand ou égal à la taille du bloc demandé
- «Next fit» : comme précédemment, mais on commence à l'endroit où on s'est arrêté (au lieu de recommencer à chaque fois au début)
- «Best fit» : On cherche le bloc dont la taille correspond au mieux à notre besoin. Cet algorithme est lent et laisse beaucoup de petits blocs inutilisables.
- «Worst fit» : On cherche le bloc le plus grand. C'est aussi lent et des simulations ont montré que ce n'est pas la bonne solution non plus.

Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)

On peut accélérer les recherches en maintenant 2 listes, une pour les blocs libres et une autre pour les blocs alloués, mais la dé-allocation sera alors moins efficace.

Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)

On peut aussi trier la liste par taille des blocs :

- Ca nous permettra d'implémenter les algorithmes «Best fit» et «Worst fit» plus efficacement.

Gestion de la mémoire libre

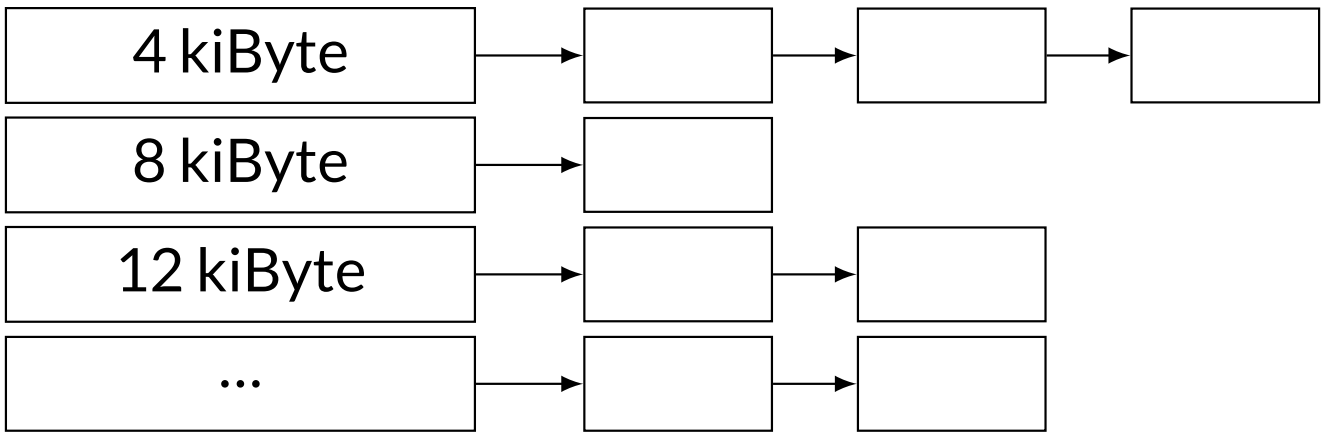
avec des listes chaînées (Linked Lists)

Si on utilise deux listes, on peut utiliser les trous eux-mêmes pour les lier entre eux. Il suffit de stocker les pointeurs («prev» et «next») ainsi que la taille directement dans le trou. Mais ça signifie que les trous ne peuvent pas être trop petits (3 × 4 Byte = 12 Byte sur un système 32 bit traditionnel)

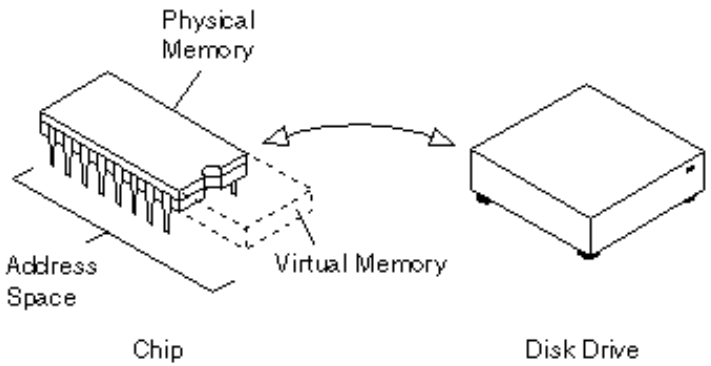
Gestion de la mémoire libre

avec des listes chaînées (Linked Lists)

Il existe encore un autre algorithme pour la recherche de blocs libres, le «Quick fit». L'idée est de maintenir plusieurs listes, en fonction de la taille des blocs qu'elle contient :

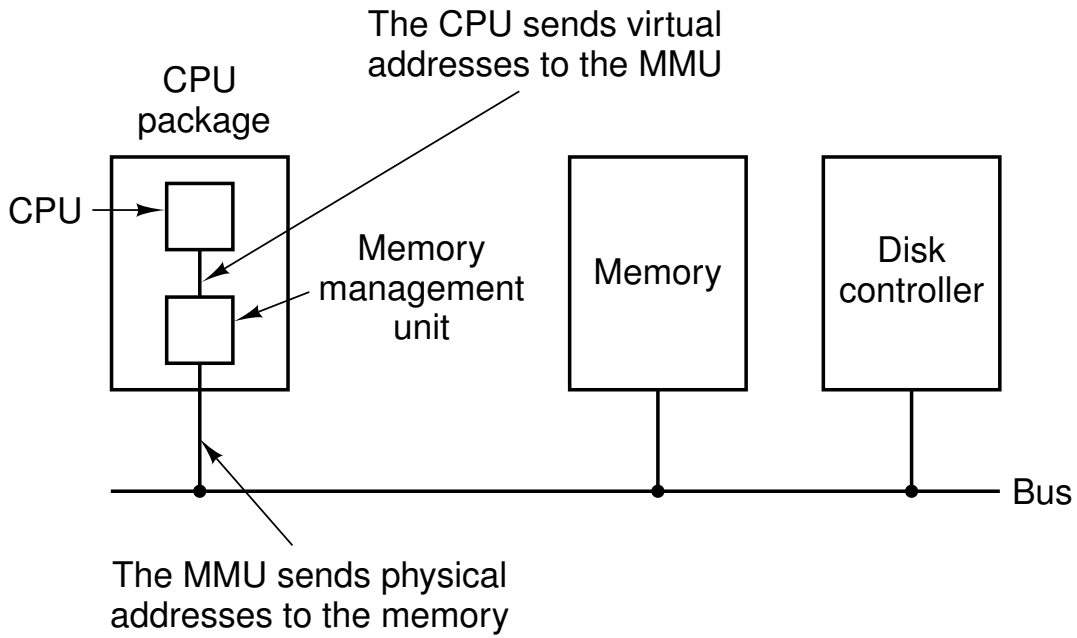


Mémoire virtuelle



Credit : *Webopedia*

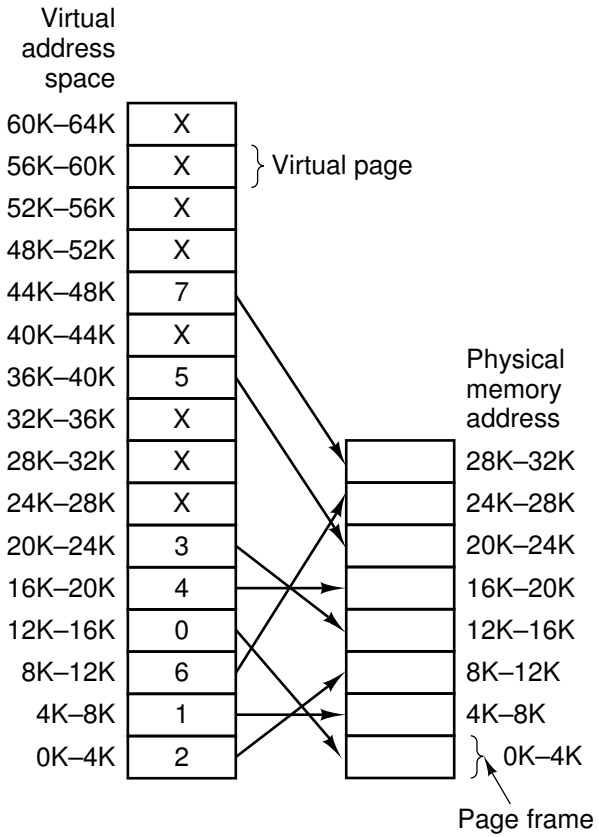
- La mémoire virtuelle permet la multiprogrammation (concurrence).
- La taille des logiciels augmente plus vitre que celle de la RAM. La mémoire virtuelle permet d'avoir des programmes plus grands que la mémoire physique.
- L'espace d'adressages est découpé en **pages** (suite d'adresses contiguës).
- Rappel : Chaque programme a son propre espace d'adressage.



Localisation et fonction d'une MMU. Ici, la MMU est montée comme une partie intégrante du CPU. Cette configuration est usuelle de nos jours. Toutefois, elle pourrait se trouver dans un circuit séparé, comme c'était le cas il y a quelques années.

Mémoire virtuelle

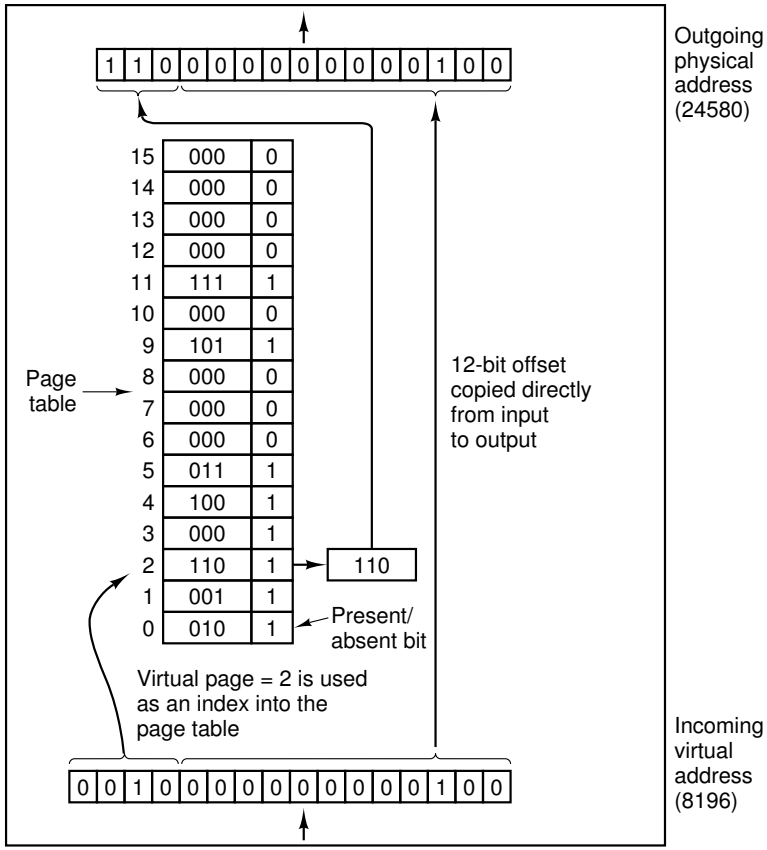
La pagination



Relation entre les adresses virtuelles (64kiB) et les adresses de la mémoire physique (32 kiB) indiquée dans la table des pages.

Mémoire virtuelle

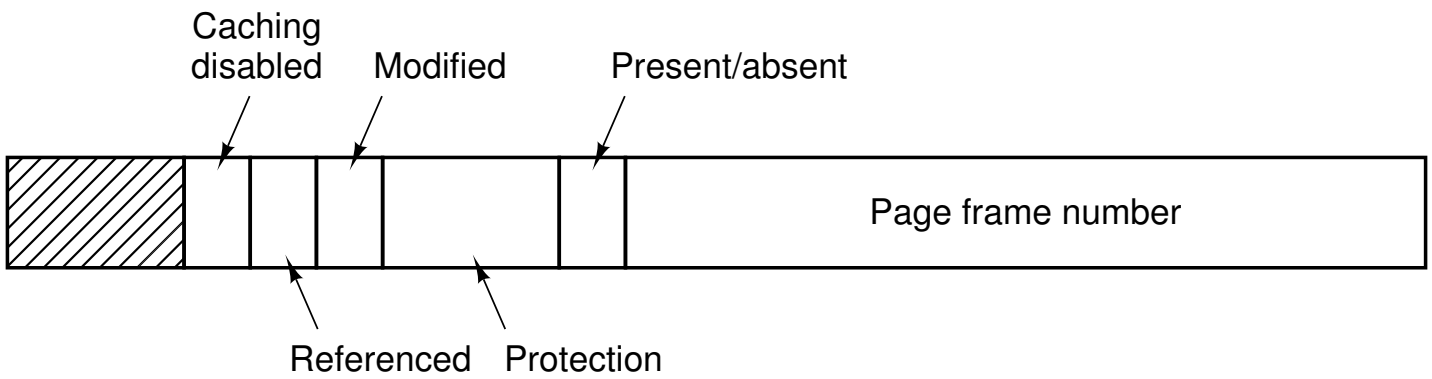
- Les pages de la mémoire virtuelle sont mappées sur la mémoire physique (cadres de page).
- Il n'est pas nécessaire d'avoir toutes les pages mappées pour exécuter un processus.
- La multiprogrammation profite des I/O générées par le «swapping» pour donner des ressources CPU à d'autres processus.



Fonctionnement interne d'une MMU avec 16 pages de 4 kiB.

Structure d'une entrée de table de pages

Entrée type d'une table de pages



- Comme il y a plus de pages virtuelles que de cadres de page (page frames), un drapeau (flag) indique si la page virtuelle est actuellement mappée à un cadre de page ou non (present bit).
- La protection peut être simple (R/RW) et un bit suffit, ou plus complexe (R/W/X) et nécessite alors 3 bits.

Mémoire virtuelle

Si le CPU essaye d'accéder à une adresse d'un page virtuelle absente :

- Une exception de type **«page fault»** est produite
- Le système d'exploitation sélectionne un cadre de page peu utilisée
- il sauve le contenu ce cette page peu utilisée sur le disque
- Charge la nouvelle page depuis le disque
- Modifie le «mapping»
- Et reprend l'instruction

Mémoire virtuelle

Détail du traitement des «page faults» (1/4)

1. Le matériel effectue un déroutement (trap) dans le noyau, en sauvegardant le compteur ordinal sur la pile. Sur la plupart des machines, certaines informations concernant l'état de l'instruction courante sont enregistrées dans des registres particuliers du processeur.
2. Une routine en assembleur est exécutée pour sauvegarder les registres généraux et d'autres informations volatiles afin d'éviter que le système d'exploitation ne les détruise. Cette routine appelle le système d'exploitation en tant que procédure.

Mémoire virtuelle

Détail du traitement des «page faults» (2/4)

3. Le système d'exploitation détecte le défaut de page et essaye de déterminer la page virtuelle manquante. Cette information est souvent contenue dans l'un des registres matériels. Dans le cas contraire, le système d'exploitation doit rétablir le compteur ordinal et parcourir l'instruction de manière logicielle afin de déterminer ce qui a provoqué le défaut de page.
4. Lorsque l'adresse virtuelle qui a causé le défaut de page est connue, le système d'exploitation vérifie la validité de celle-ci et sa cohérence en fonction des protections. En cas de problème, il envoie un signal au processus ou le tue. Si cette adresse est valide et s'il n'y a pas de défaut de protection, le système d'exploitation essaye de trouver une case mémoire libre. Si aucune case n'est disponible, l'algorithme de remplacement de page est exécuté pour en sélectionner une.

Mémoire virtuelle

Détail du traitement des «page faults» (3/4)

5. Si la case retenue a été modifiée, la page est ordonnancée pour être transférée sur le disque, une commutation de contexte suspend le processus qui a généré le défaut de page et exécute un autre processus jusqu'à la copie complète sur le disque de la page modifiée. Dans tous les cas, la case est marquée comme occupée pour éviter qu'elle ne serve à d'autres fins.
6. Dès que la case mémoire est libre (soit immédiatement, soit après son écriture sur le disque), le système d'exploitation recherche l'adresse sur le disque de la page requise et active sa lecture sur le disque. Pendant le chargement de la page, le processus responsable du défaut de page est toujours suspendu et un autre processus est éventuellement exécuté.

Mémoire virtuelle

Détail du traitement des «page faults» (4/4)

7. Lorsque l'interruption disque signale l'arrivée de la page, les tables des pages sont mises à jour pour indiquer sa position et la case est marquée comme normale.
8. L'instruction fautive est remise dans son état original et le compteur ordinal la pointe à nouveau.
9. Le processus qui a provoqué le défaut de page est ordonnancé et le système d'exploitation retourne à la routine en langage d'assemblage qui l'a appelé.
10. Cette routine restaure les registres et les autres informations, puis retourne à l'espace utilisateur pour poursuivre l'exécution, comme si aucun défaut de page ne s'était produit.

Accélérer la pagination

Tout système de pagination doit considérer ces deux éléments :

- La correspondance de l'adresse virtuelle vers l'adresse physique doit être rapide, car elle est utilisée à chaque accès mémoire (donc très souvent).
- Si l'espace d'adressage est grand, la table des pages sera grande. Pour un espace d'adressage sur 32 bit et des pages de 4 kiByte, il y a 1'000'000 de pages !

- De plus, chaque processus à besoin de sa propre table de pages !

Accélérer la pagination

2 modèles possibles

- Un MMU rapide avec copie en RAM. Chaque changement de contexte conduit à un chargement/déchargement complet du MMU
- La table est entièrement en mémoire et le MMU se résume à un registre qui pointe vers le début de cette table.

Mais aucun de ces modèles n'est vraiment efficace.

Translation Lookaside Buffer (TLB)

Observation : Les références dans la table sont assez *groupées* et seule une petite partie de la table est effectivement utilisée.

Translation Lookaside Buffer (TLB)

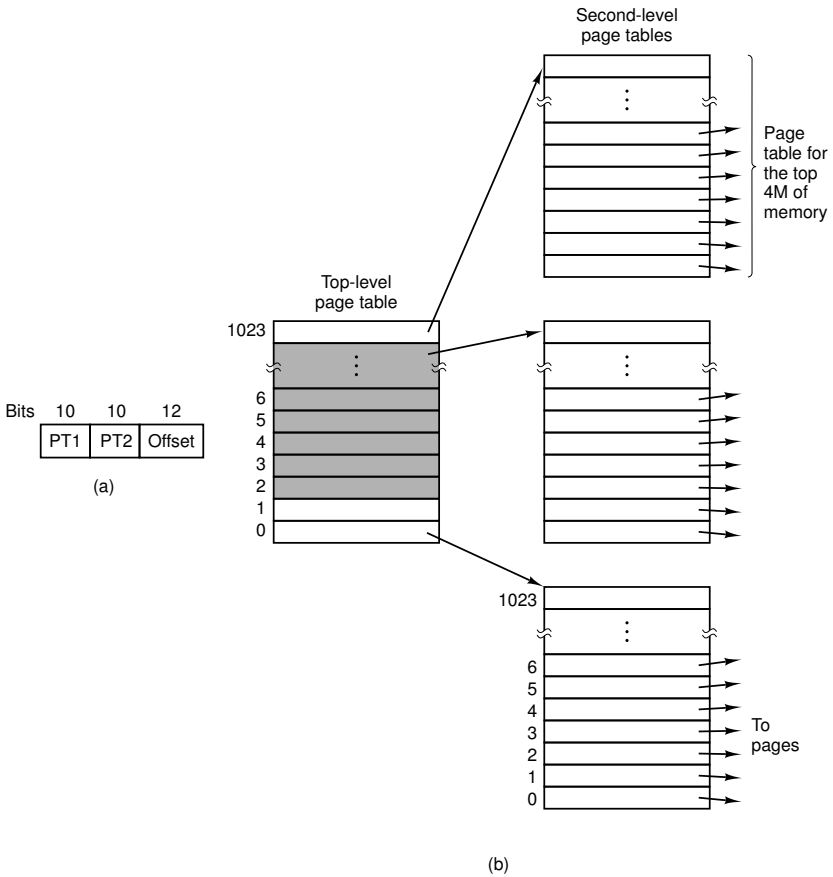
le TLB accélère la pagination

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

- Le TLB est une petite table (typiquement < 64 entrées) très rapide. La table principale reste en RAM.
- Lorsque le numéro de la page virtuelle n'est pas dans le TLB, le MMU remplace une des entrées par celle de la table des pages en RAM.

Table de pages pour les très grandes mémoires

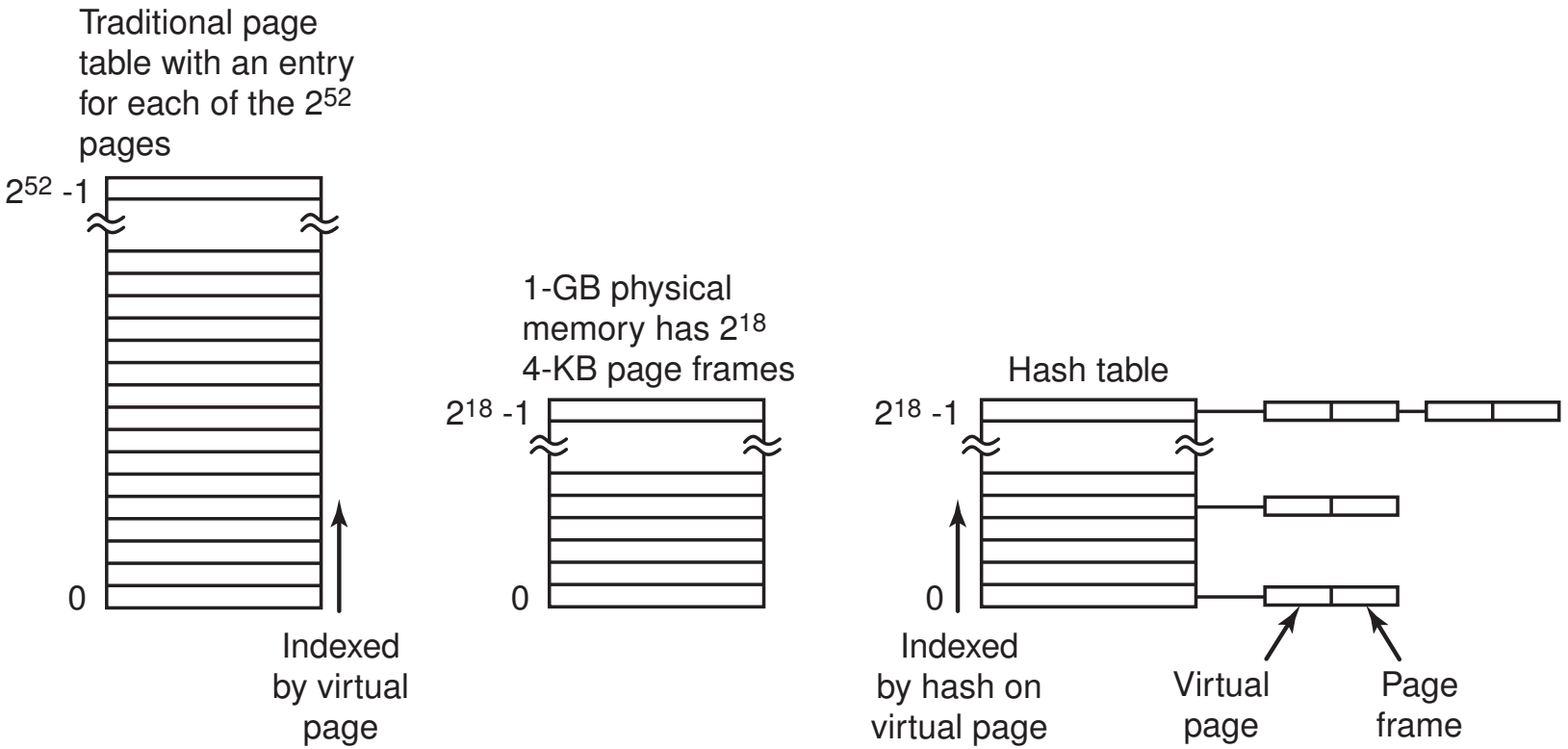
Tables des pages multi-niveaux



- (a) Une adresse 32 bits avec deux champs de table des pages
- (b) Tables des pages à deux niveaux.

Table de pages pour les très grandes mémoires

Tables des pages inversées



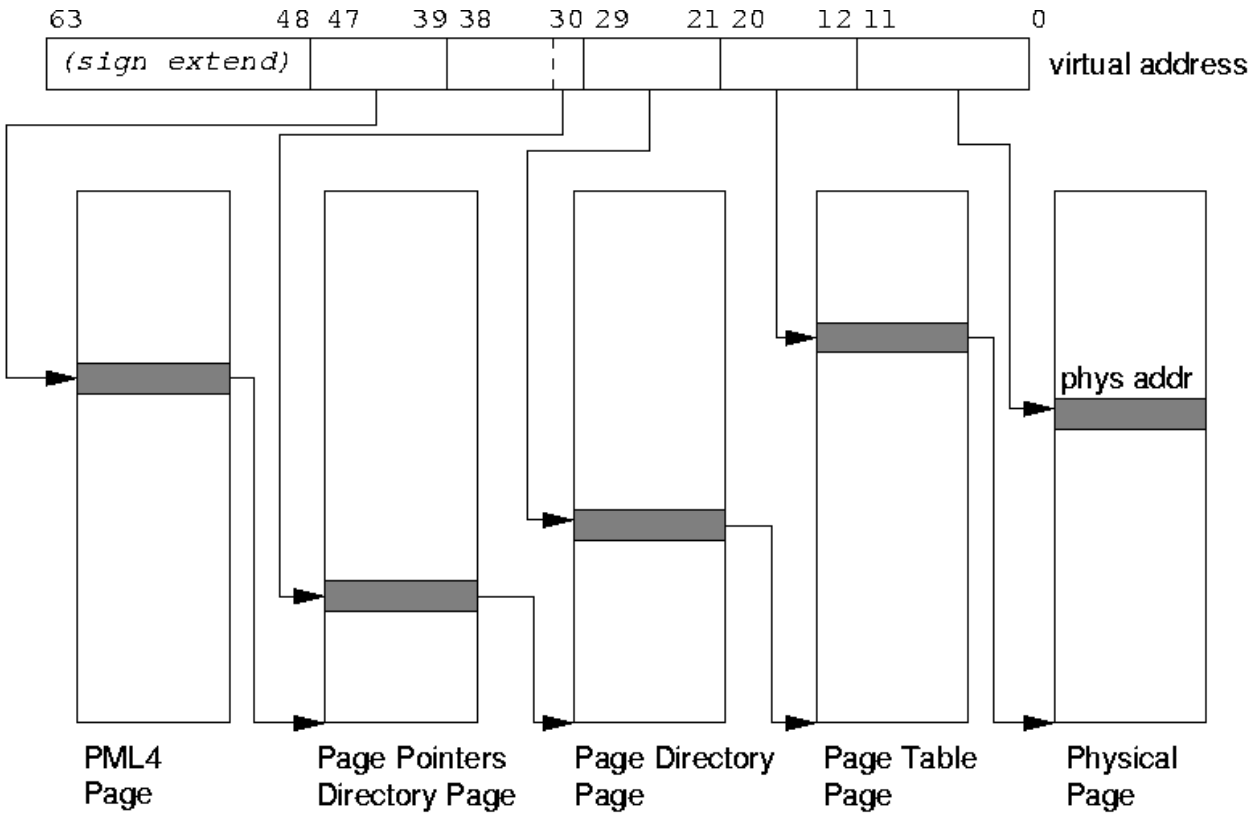
Comparaison entre une table des pages traditionnelle et une table des pages inversée.

Table de pages pour les très grandes mémoires

Tables des pages inversées

- Avec une entrée dans la table par cadre de page physique, on économise beaucoup de place. Par exemple, un système avec un espace d'adressage de 64 bit, un RAM de 1 GiByte, et des pages de 4 kiByte aura besoin de $\frac{1\ GiB}{4\ kiB}$, soit 262'144 entrées.
- On économise de la place, mais on perd du temps. Il faut en effet maintenant «chercher» la bonne page dans la table !
- solutions : TLB et/ou utilisation d'une table de hachage.

Exemple : Le processeur x86-64 (AMD64)



Credit : *Frank van der Linden, Wasabi Systems, Inc.*

Pages de 4 kiByte, 4 niveaux, tables de 512 entrées $\rightarrow$ espace d'adressage de 256 TiByte.