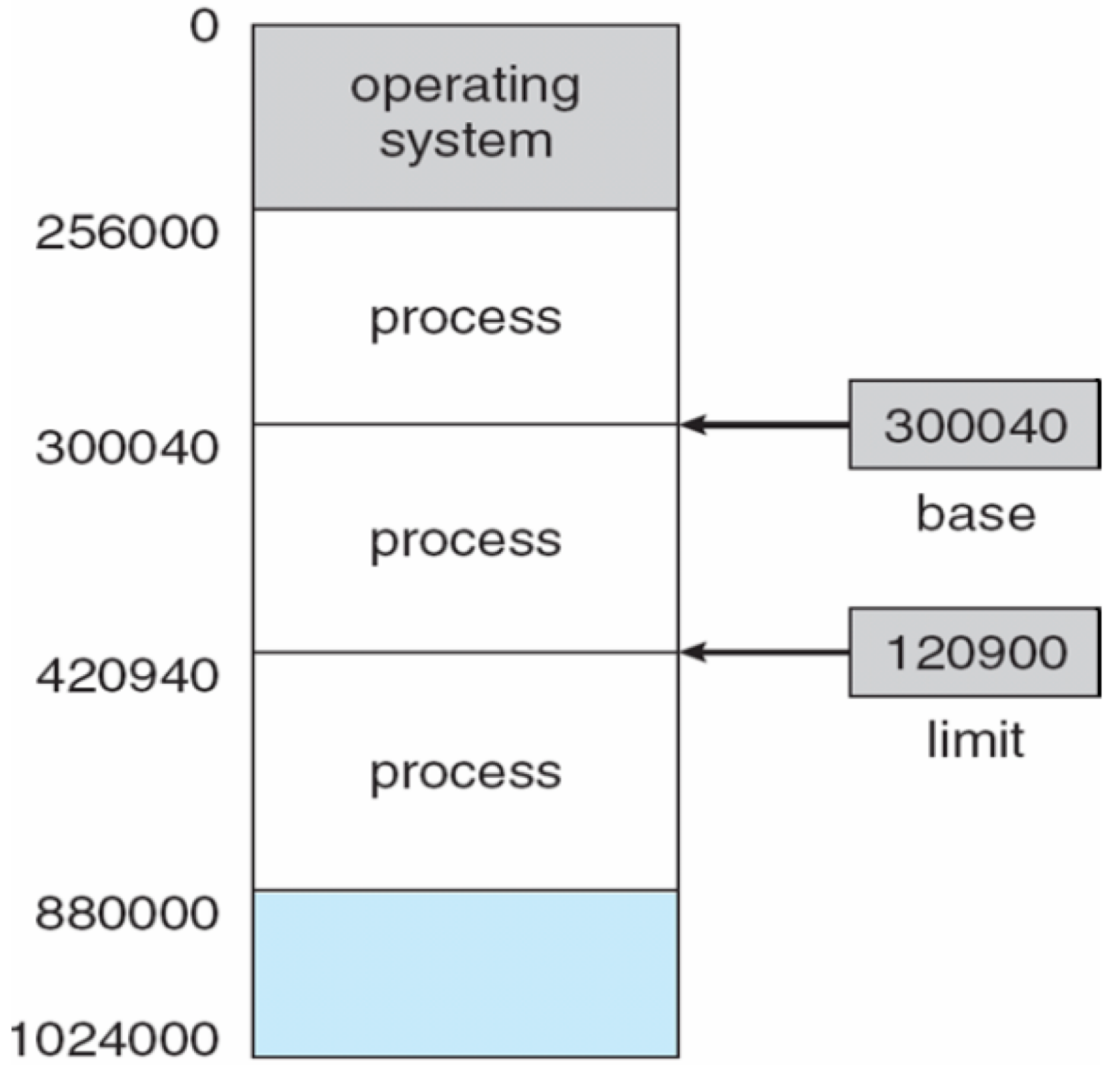
Mémoire Principale

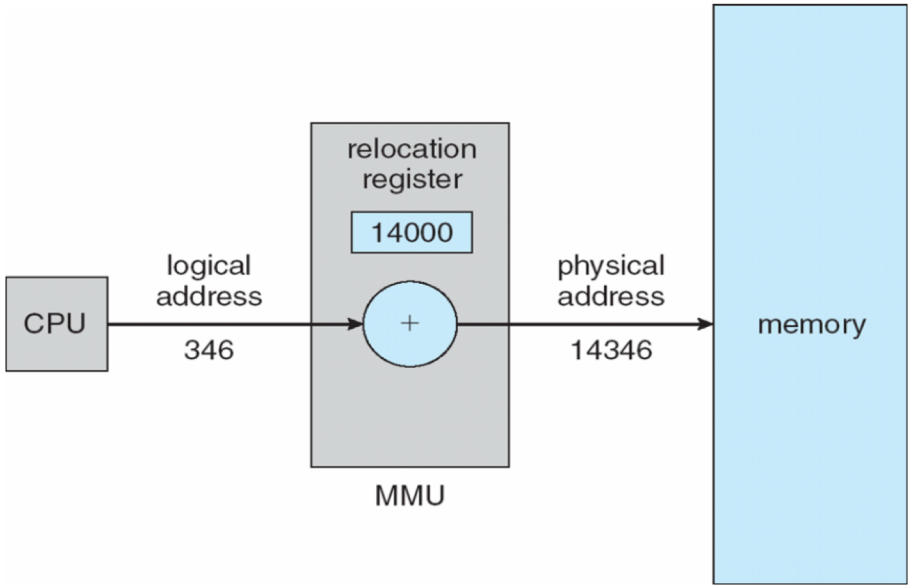
Démonstration

- Chaque processus reçoit un segment de mémoire, contrôlé par le noyau via le registre de base et limite

- Le système d'exploitation a un accès complet pour gérer les allocations de mémoire, les processus, et assurer la sécurité.



Adresses Logiques vs. Physiques



- **Logique** : Vue du processus sur sa portion de mémoire.
- **Physique** : La vraie adresse dans la mémoire globale.

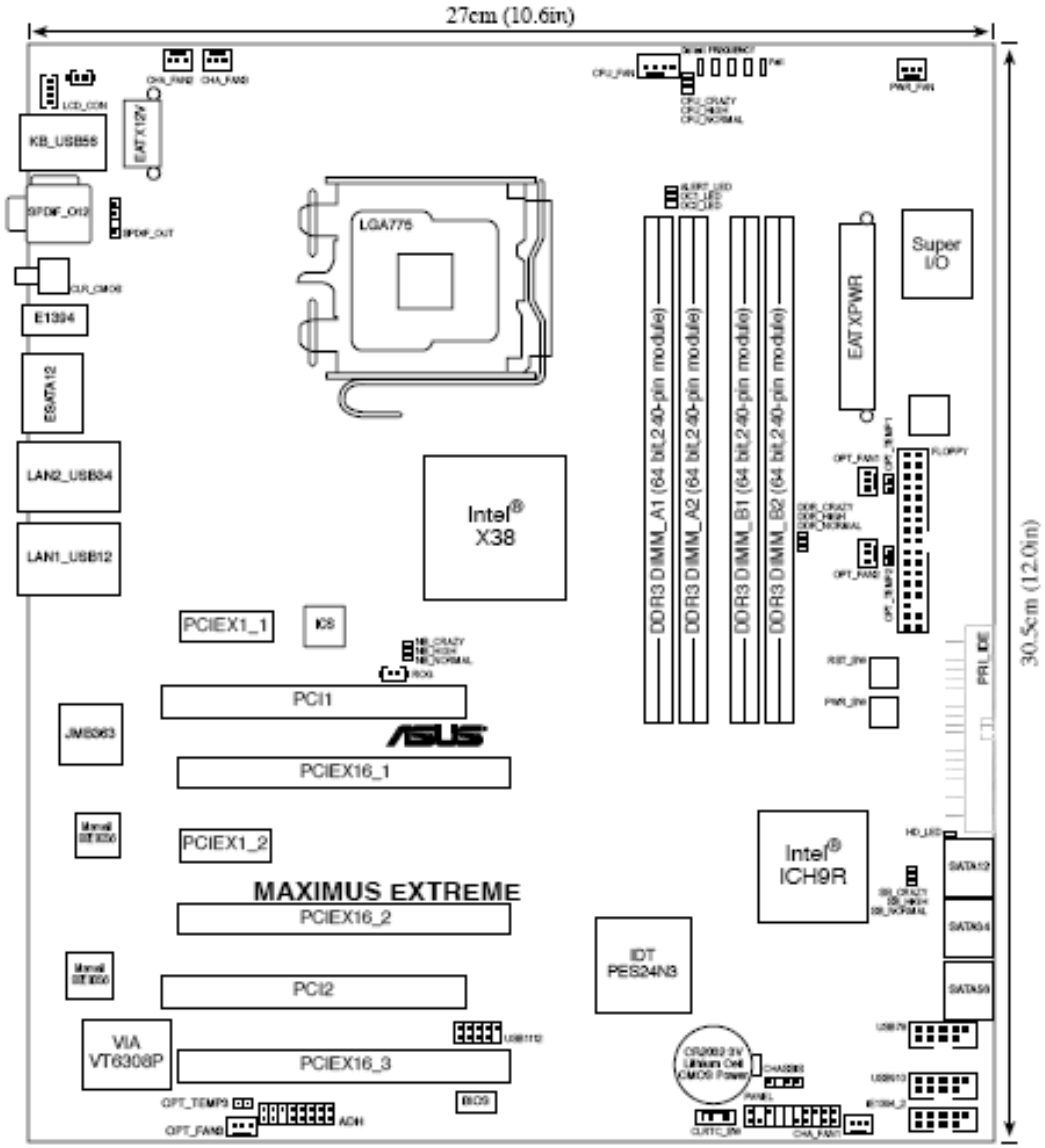
- **Isoler l'espace des processus**

<u>Rôle du MMU</u>

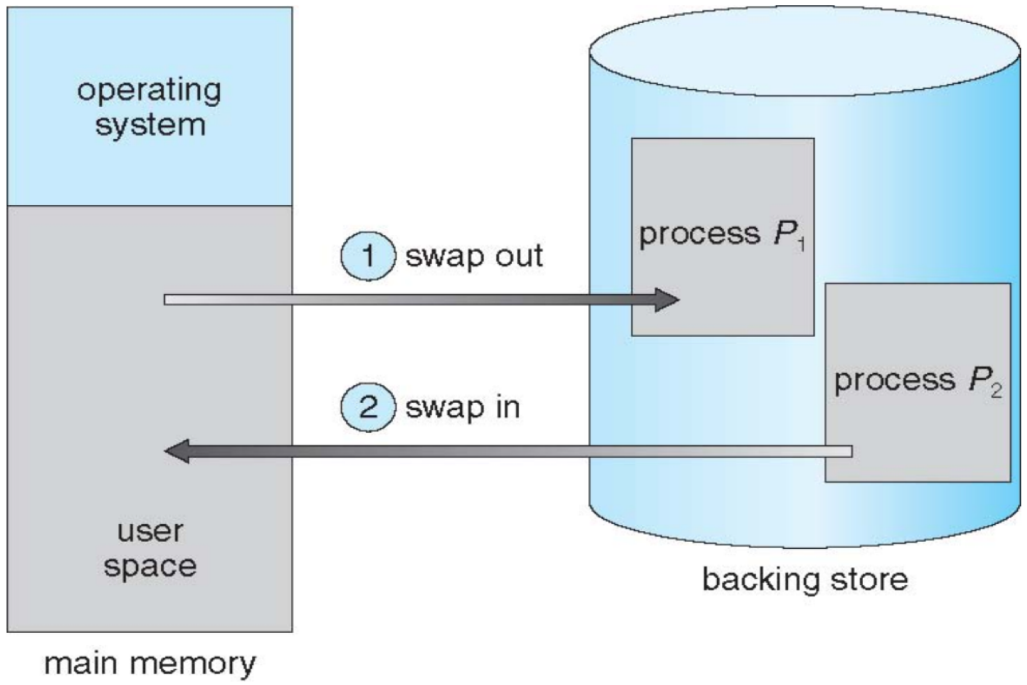
- Vérifie l'accès à l'espace alloué pour chaque processus.
- Traduit les adresses logiques en physiques, permettant une isolation efficace des processus.

Mémoire Physique vs. RAM

- La mémoire physique inclut la RAM et d'autres types de mémoire directement accessibles par le CPU.
- Importance de la distinction entre mémoire physique et virtuelle pour la gestion efficace de la mémoire.
- RAM est physiquement proche du processeur pour réduire la latence et augmenter le débit de données.



Swapping

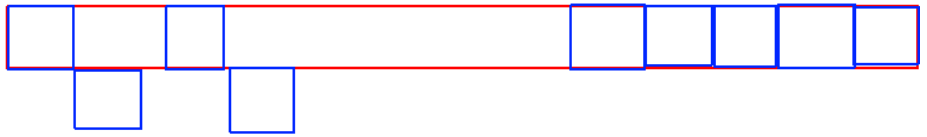


- Mécanisme de swapping pour optimiser l'utilisation de la RAM
- Importance d'avoir une quantité suffisante de RAM (mémoire physique) pour minimiser les performances lentes dues au swapping

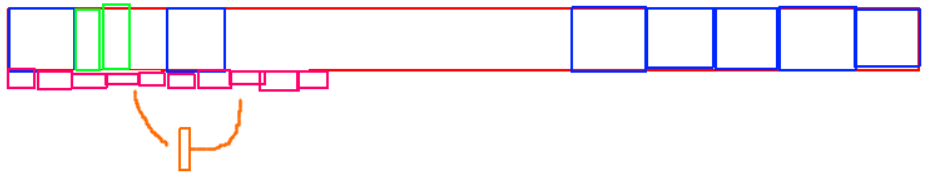
Long!!

Gestion de Mémoire Contiguë

Worst fit: bon si tous les processus sont ~ aussi grands. Parce que best-fit fait de la fragmentation interne.



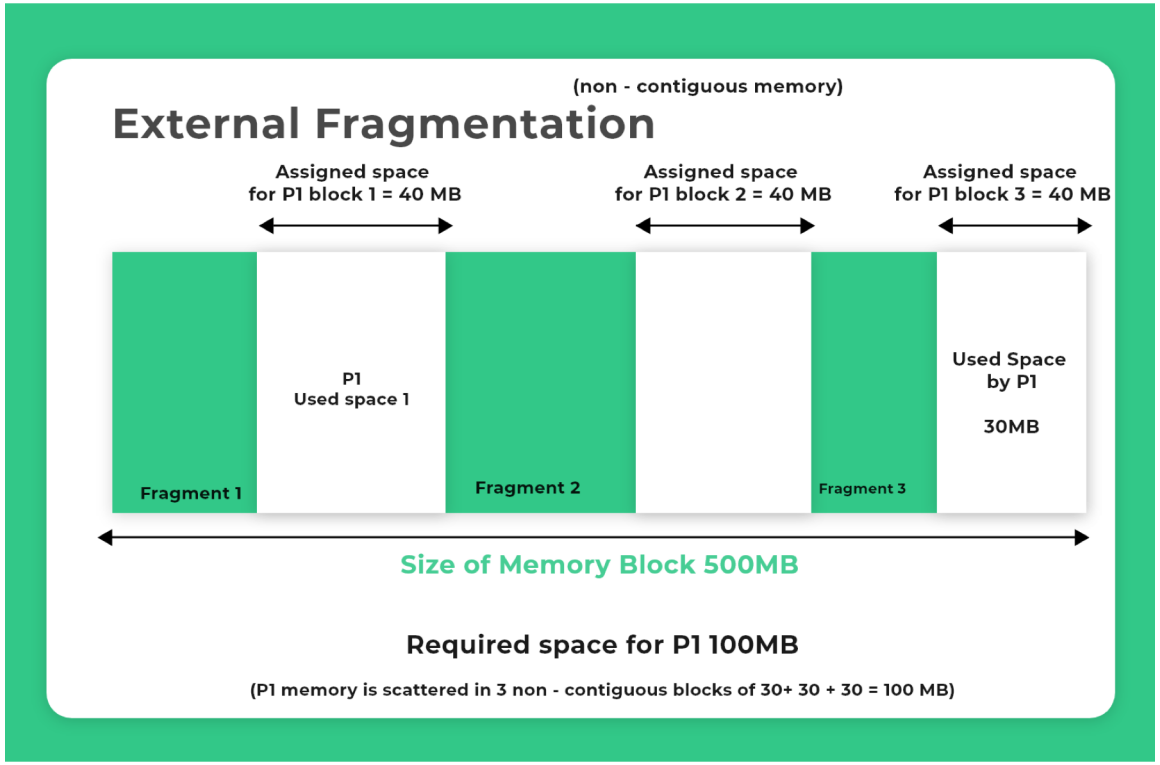
Best fit: utilise les trous de façon “optimale”, bien lorsque les processus sont de taille variable, mais peut faire de la fragmentation interne.

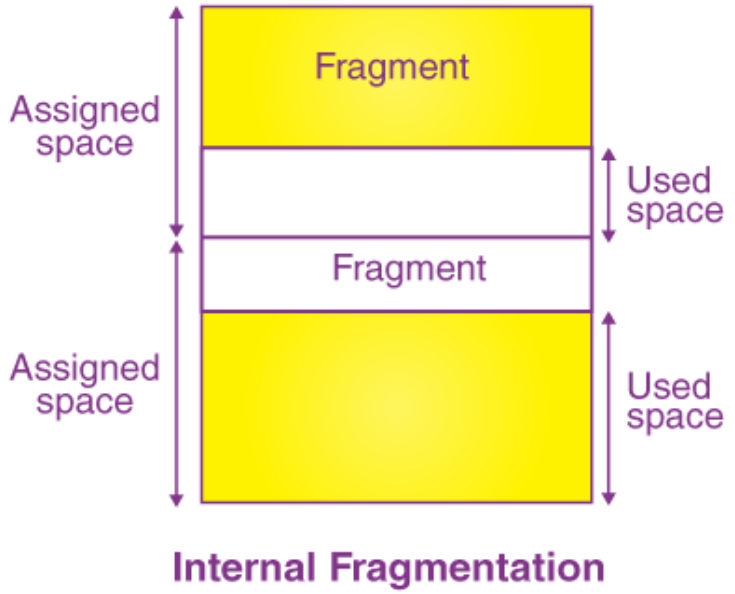


First fit (ouga bouga programmeurs aime easy greedy): Étonnamment pas terrible, puisque ça concentre les petits programmes vers le haut de la mémoire, menant vers une hiérarchie de grosseur de programmes naturelle.

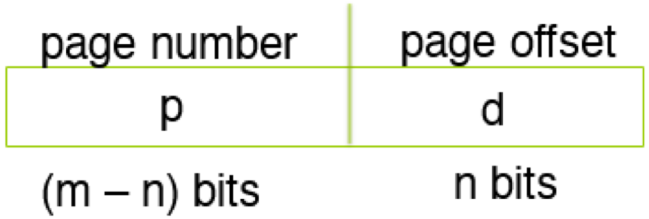
Fragmentation

- **Externe**: je place la forme dans un trou pas continu à d'autres formes. Donc je viens de diviser plein de trous collés en deux. (partition variable)
- **Interne**: ma forme est petite et je la place dans un plus trou, laissant pas assez de place pour placer une autre forme. (partition fixe)





Pagination

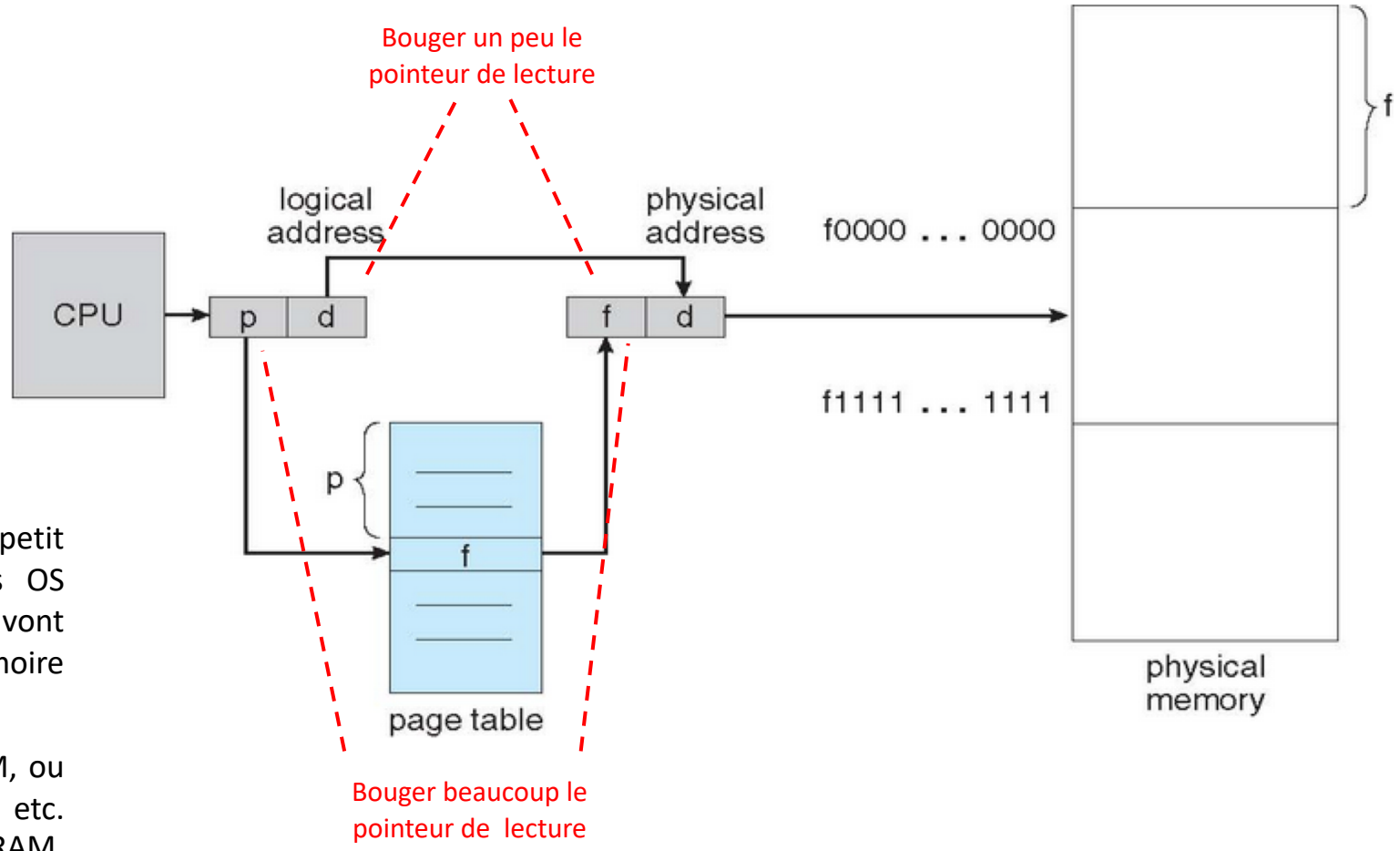


- Système qui évite la fragmentation externe en divisant la mémoire en pages et cadres de taille fixe.
- La division en pages facilite la traduction d'adresses et optimise la gestion de la mémoire.

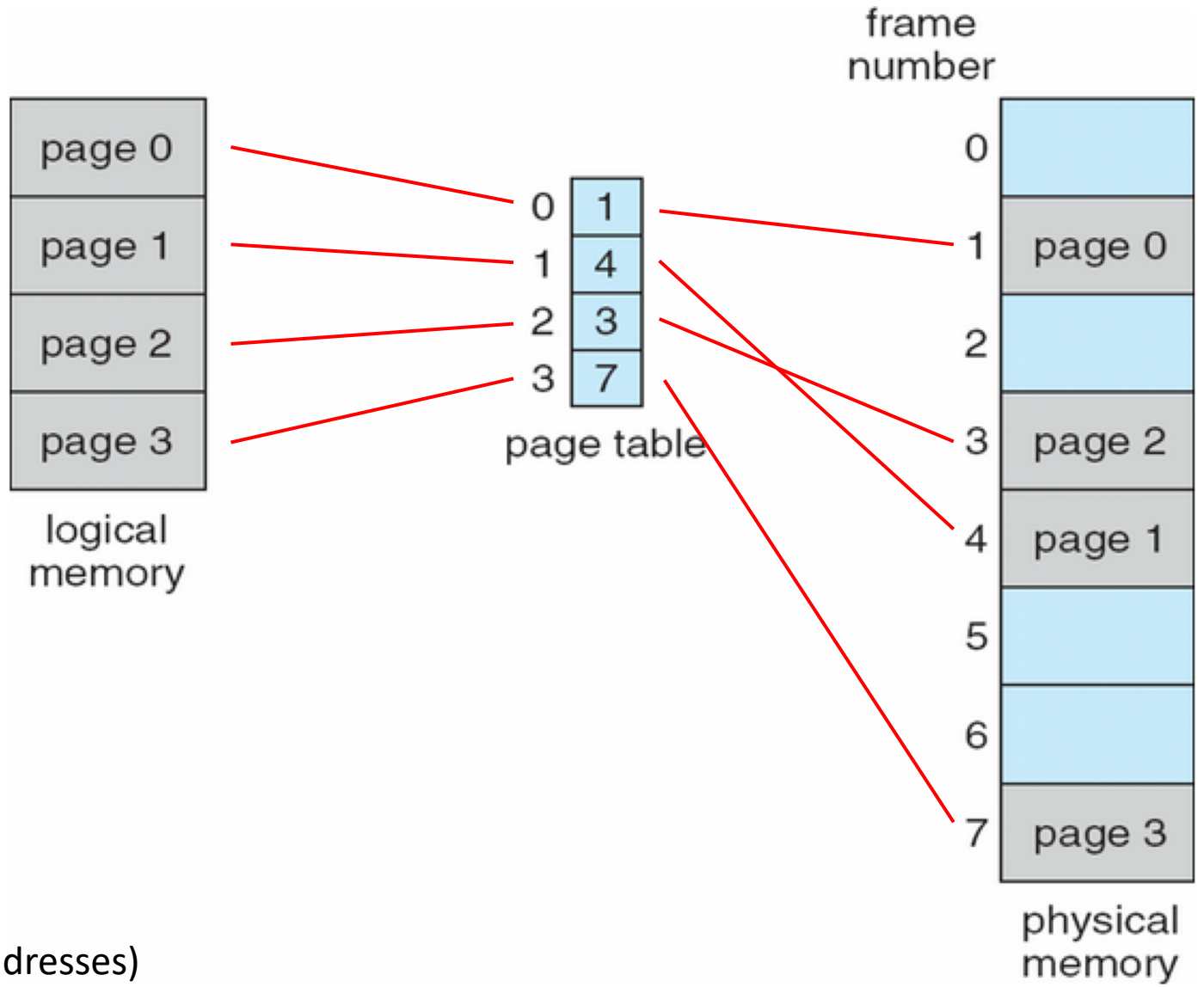
Comment les technologies émergentes telles que la réalité virtuelle (VR) et la réalité augmentée (AR) remettent-elles en question les stratégies actuelles de gestion de la mémoire ? 👀

Table de Pages

- Chaque processus dispose d'une table de pages pour le mappage des pages virtuelles aux cadres physiques.
- Variabilité de la structure des tables de pages pour répondre aux besoins spécifiques.



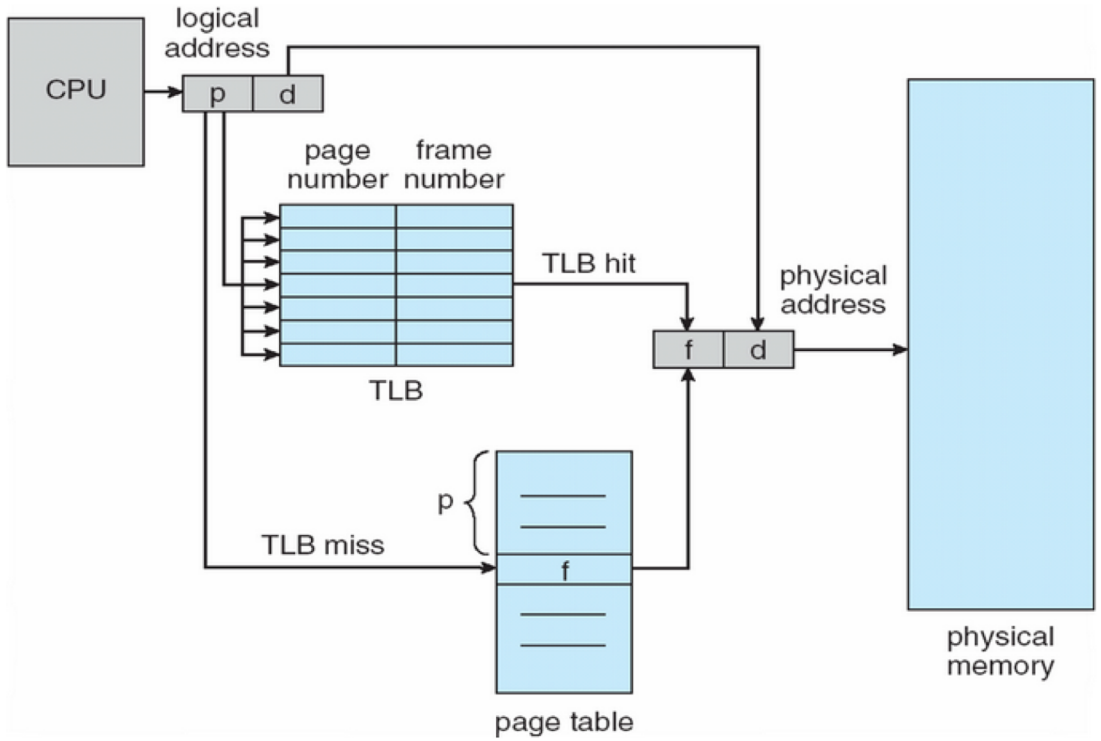
- On peut avoir un espace logique énorme pour un très petit espace physique -> standardiser nos OS: tous les OS compatibles avec des adresses logiques de 64 bits vont fonctionner sur n'importe quelle quantité de vraie mémoire physiques.
- C'est pour ça qu'on peut ajouter ou enlever de la RAM, ou qu'un Chromebook (ew) de 4GB, un android de 2GB, etc. peut rouler le même kernel qu'un ordi avec 128GB de RAM, etc.



(Comme un livre d’adresses)

TLB

- Mémoire cache pour accélérer la traduction d'adresses virtuelles en physiques en stockant les mappages récents.
- Réduit la latence de traduction d'adresse grâce à un accès rapide aux traductions fréquemment utilisées.
- Augmente l'efficacité en minimisant les accès à la mémoire principale pour la traduction d'adresse.



ε : Temps de cherche dans le TLB

α : TLB hit ratio (% de temps que la frame est trouvée dans le TLB)

m: Temps d'accès à la mémoire

- **Temps d'accès effectif** (*Effective Access Time*) (EAT)

$$EAT = (\varepsilon + 1m)\,\alpha + (\varepsilon + 2m)(1 - \alpha)$$

$$= 2m + \varepsilon - m\,\alpha$$

Exos!

Final 2019:

Les détails suivants s'appliquent aux questions (b) - (d): Étant donné un système avec un espace d'adressage de 64 bits, une taille de page de 1 MB (2^{20} B) et avec la taille d'une entrée dans la table de pages de 8B (=64b):

(b) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(c) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page), combien de niveaux de pagination sont nécessaires et comment l'espace d'adressage est-il divisé ?

(d) (1 points) Considérant qu'un programme a besoin de 1GB (2^{30} B) d'espace mémoire virtuel. combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (c).

Exos!

Final ~~2023 (SIKE~~!) 2020:

Les détails suivants s'appliquent aux questions (c) - (f): Étant donné un système avec un espace d'adressage de 64 bits, une taille de page de 2MB et avec la taille d'une entrée dans la table de pages de 64b

(c) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(d) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page),

i. (0.5 points) combien de niveaux de pagination sont nécessaires ?

ii. (0.5 points) combien de bits sont utilisés pour la table de pages la plus externe (*outer*) ?

(e) (1 points) Considérant qu'un processus avec 128MB stocké dans la memoire centrale, combien de rangée dans la table de page d'un niveau décrite en (c) vont être marquées "valide" ?

(f) (1 points) Considérant qu'un programme a besoin de 64GB d'espace mémoire virtuel, combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (d).