Université Mohammed V- Rabat
École Mohammadia d'Ingénieurs
Département Génie Informatique

Filière Génie Informatique et Digitalisation

SYSTÈMES D'EXPLOITATION 2

Mr N. EL Faddouli

elfaddouli@emi.ac.ma, nfaddouli@gmail.com

2024-2025

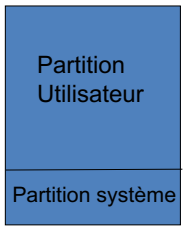
GESTION DE LA MÉMOIRE

Gestionnaire de la mémoire

- La mémoire est une ressource gérée par le **Gestionnaire de la Mémoire** qui permet de:
- Connaître les parties libres
- Connaître les parties occupées
- Allouer de la mémoire aux processus qui en ont besoin
- Récupérer la mémoire utilisée par un processus qui s'est terminé
- Gérer le va et vient (swapping) entre la mémoire secondaire et la mémoire centrale, …
- Deux modes principaux de gestion de la mémoire:
- Les systèmes à mémoire réelle
- Les systèmes à mémoire virtuelle

Systèmes à mémoire réelle (1/6)

1) Système mono-programmé

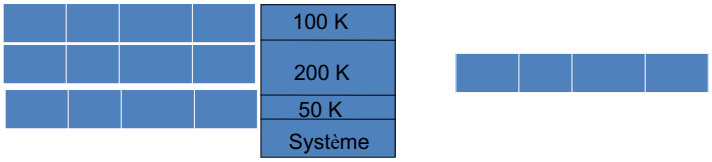


Utilisation d'un registre limite pour protéger la partition système

- Mauvaise utilisation de la mémoire (*un seul processus*) et du processeur (*attente des E/S*)
- Les possibilités du système sont limitées: les programmes sont limités à la mémoire existante

Systèmes à mémoire réelle (2/6)

2) Multiprogrammation avec des partitions fixes



- A l'initialisation du système, la **mémoire est découpée** en plusieurs **partitions de taille différentes** mais **invariantes**.
- Des **registres limites** peuvent être utilisés pour encadrer chaque partition.
- Allocation: utilisation de **files multiples** ou une **file unique**

Problème: Gaspillage de mémoire si la taille demandée est beaucoup plus petite que celle des partitions.

Systèmes à mémoire réelle (3/6)

3) Multiprogrammation avec des partitions variables

- Le **nombre**, la **position** et la **taille** des partitions varient dynamiquement.
- **Allocation** et **libération** plus complexes.
- Problème de **fragmentation** non résolu.

➔ Tassement (compactage):

Le compactage consiste à déplacer les programmes en mémoire centrale de manière à ne créer qu'une seule et unique zone libre.

➔ Problème: Blocage de l'activité système pendant cette opération

Systèmes à mémoire réelle (4/6)

- Allocation d'espace contigu
- Représentation des zones libres avec une liste chaînée.
- Choix d'une zone libre.
- Libération d'une zone occupée.
- Traitement des cas où il n' y a pas de zone libre de taille suffisante.

Systèmes à mémoire réelle (5/6)

❑ Choix d'une zone libre (Allocation)

Trois stratégies pour allouer n octets:

- First-Fit
- Best- Fit
- Worst-Fit

❑ Libération d'une zone

Pour limiter la fragmentation, on fusionne la zone libérée avec la ou les zone(s) adjacentes libres (si elles existent)

Systèmes à mémoire réelle (6/6)

- Inconvénients de l'allocation contiguë:
  - Perte d'espace due à la fragmentation de la mémoire.
  - Perte de temps due au compactage.
  - Nécessité d'une stratégie d'allocation.
  - Représentation complexe de la mémoire.

Mécanisme de pagination (1/2)

Allocation d'espaces en blocs

• La mémoire logique (*l'espace d'adressage du programme*) est divisée en blocs de tailles identiques appelés **pages.**

• L'espace de la mémoire physique est découpé en parties linéaires de même taille appelées **cadres** (ou **cases**).

• Une page est le plus petit espace allouable

➔ Eviter le compactage: Les pages allouées à un processus ne sont pas forcément contiguës

Mécanisme de pagination (2/2)

- **Simplifier la représentation**: la mémoire peut être représentée par un **vecteur de bits** où chaque bit correspond à un cadre et indique si ce cadre est libre ou occupée.
- Pour libérer une zone occupée par un programme, il suffit de marquer les cadres correspondants comme libres.
- Pour allouer une zone pour **n** pages, il suffit de marquer **n** cases, comme occupées dans la mémoire. Ces pages peuvent ne pas être contiguës.

Limitations des systèmes à mémoire réelle

- L'espace d'exécution d'un processus doit être entièrement chargé en mémoire centrale.
- La capacité de la mémoire centrale limitera la taille des programmes utilisateurs.

➔ Solution: Systèmes à mémoire virtuelle

Systèmes à mémoire virtuelle

Systèmes à mémoire virtuelle (1)

- L'espace d'adressage virtuel est divisé en **pages**. Les unités correspondantes dans la mémoire physique sont appelées **cadres de pages** (*page frame*).
- Lorsqu'un processus s'exécute, **seules certaines pages** de son espace d'exécution peuvent être présentes **en mémoire réelle**, les autres seront en mémoire secondaire.

Systèmes à mémoire virtuelle (2)

- L'exécution d'une instruction ou l'accès à une donnée n'est possible que si la page la contenant est présente dans la mémoire physique.
- Une page est chargée en mémoire à la demande quand elle est **référencée**.
- Quand la page référencée n'existe pas en M.C, une interruption de **défaut de page** est déclenchée.
- Le processus correspondant est **bloqué en attente** du chargement de la page dans un cadre libre (E/S).

Systèmes à mémoire virtuelle (3)

- Lorsque tous les cadres de la M.C sont pleins, le S.E doit choisir un dont le contenu (*une page*) sera remplacé par la page référencée et qui doit être chargée dans la M.C pour poursuivre l'exécution.
- La gestion de la mémoire virtuelle, qui se base sur le mécanisme de **pagination**, doit comprendre donc une **politique de remplacement** pour la gestion des transferts entre mémoire centrale et mémoire secondaire.

Politique de remplacement

• Au bout d'un certain temps, la mémoire centrale est saturée.

• Pour charger une nouvelle page, il va falloir transférer une autre page vers la mémoire secondaire.

Problème: Quelle page chasser de la mémoire principale?

Objectif: **Diminuer** le trafic total de pages entre la M.C et la M.S

Algorithme de Belady: OPT Algorithme (1)

- Il remplace la page qui sera utilisée le plus tardivement:
  - Etiqueter chaque page avec le nombre d'instructions qui seront exécutées avant que cette page ne soit référencée.

  → Enlever la page dont l'étiquette est la plus grande

  (*on repousse ainsi le défaut de page aussi tard que possible*)

- Algorithme **optimal**, mais **impossible à mettre en œuvre**: , il nécessite de connaître **à tout instant** la séquence **future** des demandes du programme, ce qui est évidemment impossible

- Modèle théorique pour mesurer la performance des autres systèmes.

Algorithme de Belady: OPT Algorithme (2)

Remplacer la page qui ne sera pas utilisée pendant la plus longue durée

Exemple:

4 **frames** (nombre de page en MC)
Demandes de pages: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	1	1	1	1	1	4	4
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3
			4	4	4	5	5	5	5	5	5

6 défauts de pages

Algorithme Aléatoire: Random Algorithm

La page qui sera remplacée est choisie de façon aléatoire

Algorithme FIFO (1/3)

- La page la plus ancienne (*chargée*) en M.C est remplacée
- Algorithme le plus simple

Problèmes:

- Cet algorithme peut virer des pages importantes. En effet, la page la plus ancienne peut être la page la plus utilisée.
- Cet algorithme souffre de l'**anomalie de Belady**.

Algorithme FIFO (2/3)

Exemple: Anomalie de Belady

3 frames (nombre de page en MC)
Demandes de pages: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4

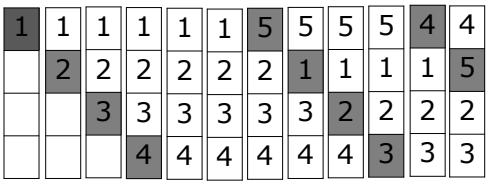
9 défauts de pages.

En général, plus il y a plus de frames, moins il y aura de défauts de pages.

Algorithme FIFO (3/3)

Revoir l'exemple avec **4 frames** au lieu de 3.

Demandes de pages: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



10 défauts de pages.

Algorithme LRU: Least Recently Used (1/2)

- La page remplacée est la **moins récemment** utilisée.
  - Un défaut de page remplace la page qui n'a pas été utilisée depuis le plus longtemps.

    (Approximation de l'algorithme optimal)

- Associer à chaque page son dernier instant d'utilisation.
  - Deux méthodes d'implémentation:
    - Avec un compteur de temps
    - Avec une pile
- Peut être implémenté au moyen d'un matériel spécialisé

Algorithme LRU: Least Recently Used (2/2)

- Implémentation avec un compteur dans la **table des pages**:
  - Chaque page a un compteur: Chaque fois que la page est référencée, l'horloge est copiée dans le compteur.
  - Quand une page doit être changée, prendre celle avec le plus vieux compteur.
- Implémentation avec une pile: à chaque fois qu'une page est référencée, elle est placée au sommet de pile.

  → Le dessus = la plus récemment utilisée

  Le fond = la moins récemment utilisée.

Exemple: 4 frames et Demandes de pages: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	1	1	1	1	1	1	5
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	4	4
			4	4	4	4	4	4	3	3	3

→ 8 défauts de pages

Algorithme LFU: Least Frequently Used (1)

- La page remplacée est la moins fréquemment utilisée.
  - Utilisation d'un compteur incrémenté à chaque fois que la page est référencée.

Exemple: 4 frames
Demandes de pages: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	3	3	5
			4	4	4	4	4	4	4	4	4

→ 7 défauts de pages

- LRU et LFU sont rarement utilisés car trop gourmands en temps de calcul et difficiles à implémenter, mais ils sont assez efficaces

Algorithme NRU: Not Recently Used (1/2)

- On remplace les pages non utilisées récemment
- Pour les déterminer, on utilise deux indicateurs d'utilisation: deux bits **R** et **M**.
- **Principe**
  - chaque fois que la page est **lue** ou **écrite** (*référencée*): **R ←1**
  - chaque fois que la page est **modifiée**: **M ←1**
  - A chaque interruption de l'horloge: **R ← 0**

  (afin de distinguer les pages récemment référencées)

Algorithme NRU : Not Recently Used (2/2)

- À un instant donnée, les pages sont organisées par classes:
  - Classe 0: non référencée, non modifiée (R= 0 et M=0)
  - Classe 1: non référencée, modifiée (R=0 et M=1)
  - Classe 2: référencée, non modifiée (R=1 et M=0)
  - Classe 3: référencée, modifiée (R=1 et M=1)
- Comment déterminer la page à virer?

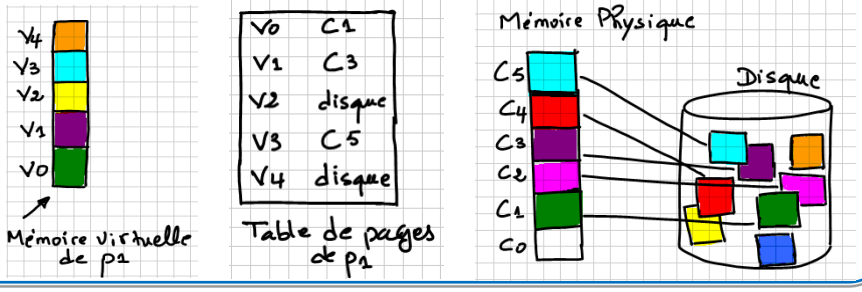
  L'algorithme **NRU** vire une page au hasard dans la plus basse classe non vide.

  Si le bit M=1 de la page à virer, elle doit être copiée sur le disque.

⇨

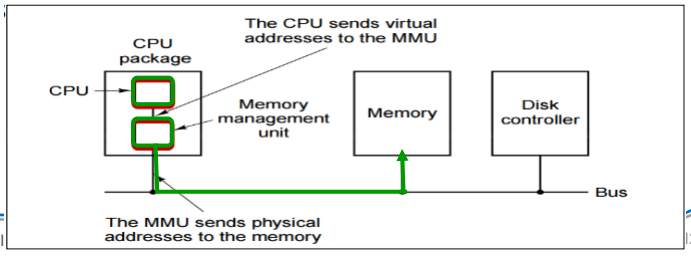
Table de pages

- Chaque processus possède sa propre **mémoire virtuelle** qui représente **l'ensemble de ses pages**.
- Il a sa propre **table de pages** qui indique la **correspondance** entre ses **pages** et les **cadres** (*frames*) qui les contiennent en **mémoire physique** (centrale).
- Chaque entrée de la table de pages contient un bit de **présence**, un bit de **référence** (**R**), un bit de **modification** (**M**), le numéro (***adresse de base***) du cadre contenant la page, **…**



Adresse Logique & Adresse physique

- Le programme utilisateur n'aperçoit pas les adresses physiques; il traite les **adresses logiques** (*générées par les compilateurs et les éditeurs de lien*) qui seront converties en adresses virtuelles par le SE au lancement du processus.
- Pendant l'exécution d'un processus, le **CPU génère des adresses virtuelles** qui sont utilisées à l'intérieur du programme.
- Une **adresse virtuelle** doit être convertie en **adresse physique** reconnue par l'**unité de mémoire**.
- **MMU** (***Memory Management Unit***) est un dispositif matériel qui fait la conversion des adresses virtuelles en adresses physiques.
- Le **MMU** était sous forme d'un chip séparé entre le CPU et la mémoire. Il est maintenant intégré au processeur.



Conversion @logique-@physique (1)

Une adresse virtuelle *(p , d)* est composée de:

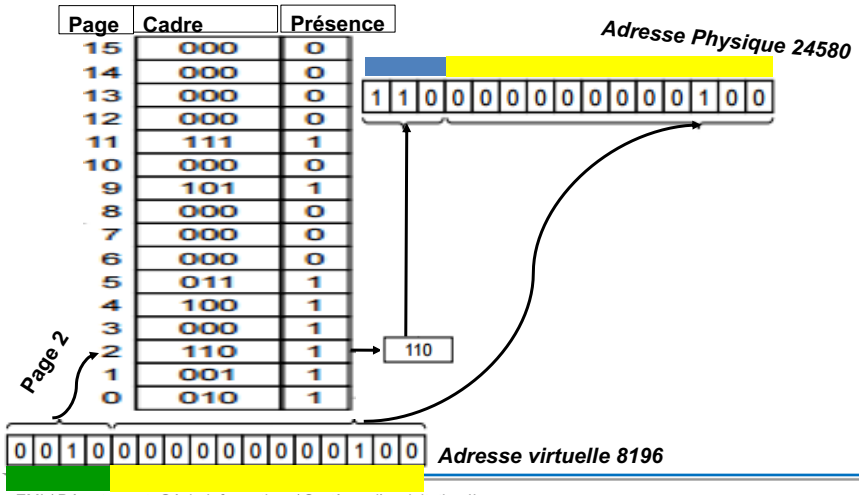
1. **Numéro de page** *p*: Indice dans la table de pages qui contient l'**adresse de base pour chaque cadre**.
2. **Déplacement** *d* **dans la page** : On le concatène avec le numéro du cadre *(adresse de base)* pour avoir l'adresse physique.

Chaque entrée dans la table des pages possède un **bit de présence** qui indique si la page est présente dans la mémoire physique ou non. Si elle est présente, le numéro du cadre qui la contienne est indiqué.

Page	Cadre	Présence
15	000	0
14	000	0
13	000	0
12	000	0
11	111	1
10	000	0
9	101	1

Conversion @logique-@physique (2)

Exemple: L'adresse virtuelle est sur 16 bits
4 bits pour le numéro de page *(⇨ 2^4=16 pages du processus)*, 12 bits pour le déplacement (offset) *(⇨ Taille de page = 2^{12} octets = 4Ko)*



Implantation des tables de pages (1)

Pour l'implantation des tables de pages, il y a deux contraintes importantes à considérer:

❑ La table des pages est extrêmement grande (plus d'un million d'entrées).

- ➢ On a des adresses d'au moins 32 bits.

  Exemple: Pour un processeur 32bits et page de 4Ko (2^{12}) on aura le nombre de pages = $2^{32}/ 2^{12} = 2^{20}$ =1 048 576

- ➢ Chaque processus a besoin de sa propre table de pages.

❑ La correspondance doit être rapide sachant que pour chaque instruction il est nécessaire de faire référence à la table des pages 1 fois, 2 fois et parfois plus.

Implantation des tables de pages (2)

Solution simple: Utilisation d'un PTBR

❑ La table de pages d'un processus est stockée dans la **mémoire centrale**.

❑ L'adresse de cette table est conservée dans le PCB du processus.

❑ Utilisation d'un registre du CPU "**Page Table Base Register**" (PTBR) qui contient l'adresse de début de la table de pages du processus en cours d'exécution (élu).

❑ A chaque commutation de contexte, le **PTBR** est modifié.

❑ **Problème**: Bien que l'utilisation de PTBR génère des changements de contexte plus rapides, la recherche de correspondance n'est pas très rapide dans la mémoire centrale.

Implantation des tables de pages (3)

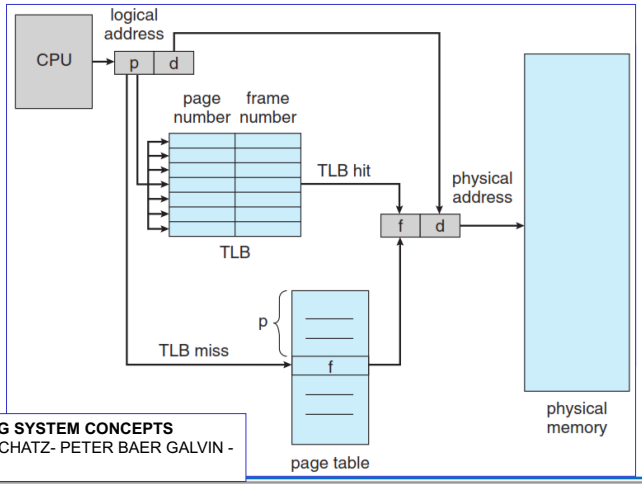
Solution standard: Translation Look aside Buffers(TLB)

- C'est un petit cache matériel spécial à recherche et consultation rapide.
- Le TLB est une mémoire **associative** à grande vitesse: chaque entrée se compose d'une **clé** et une **valeur**.
- Contiennent seulement quelques entrées de la table des pages (entre 32 et 1024).
- Certains CPU implémentent des TLB d'adresses d'instructions et de données distincts

Implantation des tables de pages (4)

Solution standard: Translation Look aside Buffers(TLB)

- Utilisation du TLB pour la conversion d'une adresse virtuelle en adresse physique:



Source: **OPERATING SYSTEM CONCEPTS** ABRAHAM SILBERSCHATZ- PETER BAER GALVIN - GREG GAGNE

Implantation des tables de pages (5)

Solution standard: Translation Look aside Buffers(TLB)

❑ Pour convertir une adresse virtuelle, le MMU cherche le numéro de cadre dans le TLB. S'il n'existe pas (**manque de TLB**):

- Le MMU l'obtient à partir de la table de pages en mémoire centrale.
- Le numéro de page et celui du cadre sont ajoutés dans le TLB pour les utiliser dans les prochains référencements.

❑ Si le TLB est **plein**, une **politique de remplacement** est utilisée (LRU, FIFO, Aléatoire, ...) pour sélectionner l'entrée à remplacer.

Implantation des tables de pages (6)

Solution standard: Translation Look aside Buffers(TLB)

❑ Certains TLB stockent l'identifiant d'espace d'adressage **ASID** (*Address-Space IDentifier*) dans **chaque entrée** TLB.

❑ Un ASID identifie de manière unique chaque processus et est utilisé pour fournir une **protection de l'espace d'adressage** pour ce processus.

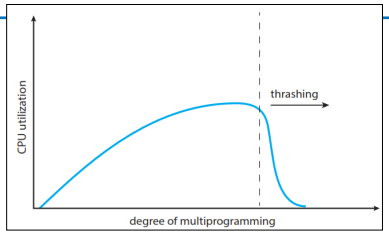
❑ Un ASID permet au TLB de contenir des entrées pour plusieurs processus différents simultanément (→ changement de contexte rapide)

❑ Si le TLB ne prend pas en charge les ASID séparés, chaque fois qu'une nouvelle table des pages est sélectionnée (*changement de contexte*), le TLB doit être vidé pour charger une partie de cette table.

Limitations de remplacement (1)

Problème d'écroulement: **Trashing**

Niveau de multiprogrammation élevé



➔Quand le nombre de processus augmente, le nombre de pages présentes pour chacun diminue

➔de plus en plus de défauts de pages et attente de pages:

trop de défauts de pages → un processus peut passer plus de temps en attente de pagination qu'en exécution : ***trashing***.

→ Performance très amoindrie du système. (CPU)↘

➔ Le système passe son temps à remplacer des pages de certains processus pour les donner aux autres.

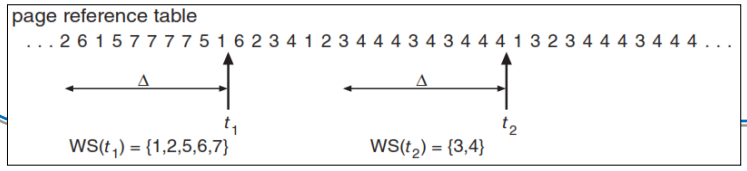
Limitations de remplacement (2)

Comment minimiser le trashing ?

• **Algorithme de remplacement local** : un processus qui fait du trashing ne peut pas prendre de frames aux autres processus.

• Fournir à un processus autant de frames qu'il en a besoin. Utiliser par exemple le ***Working Set Model***.

- **Working Set** (**Ensemble de Travail**): c'est l'ensemble des pages du processus qui doivent être chargées à un instant donné pour que le processus s'exécute *efficacement*.
- L'ensemble de travail d'un processus est déterminé en fonction d'un paramètre Δ qui détermine le nombre de référencements de page effectués dernièrement.



Limitations de remplacement (3)

- Si l'ensemble de travail est entièrement en mémoire, le processus ne fera pas beaucoup de défauts de pages.
- Le système de pagination doit s'assurer que l'ensemble de travail est en mémoire avant que le processus ne puisse avoir accès au processeur:

  ⇨ **pré-pagination**