- Un page fault survient lorsque le système d'exploitation détecte qu'une page à laquelle un processus tente d'accéder n'est pas chargée en mémoire physique :

1. Le système d'exploitation suspend le processus qui a causé le page fault.
2. Il localise la page nécessaire sur le disque.
3. Si aucune place n'est disponible en mémoire, un algorithme de remplacement de pages est utilisé pour libérer de l'espace.
4. La page est chargée en mémoire depuis le disque.
5. Le processus est repris, et l'instruction qui a causé le page fault est réexécutée.

- La pagination sur demande et le remplacement de pages sont deux concepts clés dans la gestion de la mémoire virtuelle. Ils permettent aux systèmes d'exploitation de gérer efficacement la mémoire physique limitée tout en offrant aux applications l'illusion d'avoir accès à une quantité de mémoire virtuelle quasi illimitée

- Lazy swap -> que les pages nécessaires dans la mémoire

- La pagination sur demande -> pages ne sont chargées en mémoire physique que lorsque cela est nécessaire. **Cela contraste avec des stratégies antérieures où toutes les pages d'un processus devaient être chargées en mémoire au démarrage du processus, ce qui pouvait être inefficace et gaspiller de la mémoire.**

- Quand un programme tente d'accéder à une page qui n'est pas en mémoire physique (une page "non présente"), le système génère un page fault. Le système d'exploitation doit alors charger la page demandée depuis le disque vers la mémoire physique pour que l'exécution puisse continuer.

- Le remplacement de pages est utilisé pour sélectionner une page à retirer de la mémoire physique lorsque l'espace est nécessaire pour une nouvelle page. Ceci est essentiel dans un contexte où la mémoire physique est limitée et doit être partagée entre plusieurs processus.

-> Plusieurs algorithmes existent pour déterminer quelle page retirer, chacun avec ses avantages et inconvénients.

-**Pour quoi faire ?** Ces mécanismes permettent d'optimiser l'utilisation de la mémoire physique, en s'assurant que les pages les plus fréquemment utilisées sont gardées en mémoire et que l'espace est alloué de manière efficace. Cela permet aux systèmes d'exploitation de supporter l'exécution de plusieurs processus simultanément, même si la mémoire physique totale est inférieure à la somme de la mémoire nécessaire pour tous les processus.

- Le bit valide/invalid, est un indicateur utilisé dans les entrées de la table de pages pour marquer l'état d'une page spécifique en ce qui concerne sa présence en mémoire physique. Ce bit joue un rôle crucial dans la gestion de la pagination sur demande et le processus de remplacement de pages

- **Fonctionnement** :

- Bit Valide (Présent) : Lorsqu'une entrée de table de pages a son bit valide défini (1), cela signifie que la page correspondante est chargée en mémoire physique et peut être accédée directement par le processus. Une tentative d'accès à une telle page n'entraînera pas de page fault.
- Bit Invalid (Non Présent) : Si le bit valide est réinitialisé (0), cela indique que la page correspondante n'est pas actuellement en mémoire physique. Une tentative d'accès à

une page avec un bit invalide déclenchera un page fault, obligeant le système d'exploitation à charger la page depuis le disque vers la mémoire physique.

- Le **dirty bit** est utilisé pour indiquer si une page en mémoire a été modifiée (écrite) depuis qu'elle a été chargée en mémoire ou depuis le dernier cycle de vérification.

- Lorsqu'une page est chargée en mémoire depuis le disque, son dirty bit est initialisé à 0 (non modifié).
- Si le processus modifie la page (par exemple, en écrivant de nouvelles données), le système d'exploitation définit le dirty bit de la page à 1 (modifié).
- Lorsque le système doit évincer une page de la mémoire physique, il vérifie le dirty bit de cette page :
    - Si le dirty bit est à 0 (la page n'a pas été modifiée), la page peut simplement être retirée sans plus d'action, car son contenu correspond toujours à ce qui est stocké sur le disque.
    - Si le dirty bit est à 1 (la page a été modifiée), le contenu de la page doit d'abord être écrit sur le disque (pour s'assurer que les modifications ne sont pas perdues) avant que la page puisse être retirée de la mémoire.

Importance du Dirty Bit

- **Optimisation de l'écriture sur disque** : Le dirty bit aide à réduire les opérations d'écriture sur disque inutiles. Sans cela, le système d'exploitation pourrait être contraint d'écrire chaque page évincée sur le disque, quelle que soit la nécessité, car il ne disposerait d'aucune manière de savoir si la page a été modifiée. Écrire uniquement les pages modifiées économise du temps et des ressources système, car les accès disque sont parmi les opérations les plus coûteuses en termes de performance.
- **Intégrité des données** : Assure que toutes les modifications apportées à la mémoire virtuelle par les processus sont correctement sauvegardées sur le disque, permettant une récupération des données en cas de perte de puissance ou d'autres interruptions du système.
- **Gestion efficace de la mémoire** : En permettant au système d'exploitation de distinguer entre les pages modifiées et non modifiées, le dirty bit contribue à une gestion plus efficace de la mémoire virtuelle et physique, améliorant ainsi la performance globale du système.

- **Rôle dans le Remplacement de Pages** : Lorsque le système d'exploitation doit charger une page en mémoire mais que la mémoire est pleine, il doit choisir une page à évincer pour faire de la place. Le processus de sélection ne dépend pas directement du bit valide/invalid ; cependant, ce bit est essentiel pour gérer l'accès aux pages et identifier les pages actuellement en mémoire. Les algorithmes de remplacement de pages s'appuient sur d'autres indicateurs (comme les bits d'accès ou de modification) et des stratégies (telles que LRU, FIFO, etc.) pour décider quelle page remplacer.

- Processus de Remplacement :

1. Système identifie les pages en mémoire comme candidates au remplacement. Toutes les pages avec le bit valide défini sont potentiellement remplaçables.
2. Algorithme de remplacement de pages (par exemple, LRU) détermine quelle page parmi les candidates est la meilleure à remplacer, souvent en se basant sur des critères comme la fréquence et la récence d'accès.

3. Système met à jour les bits appropriés dans la table de pages. Le bit valide de la page évincée est réinitialisé (marqué comme invalide), et si des modifications ont été apportées à la page, elle doit être écrite sur le disque avant l'éviction.
4. La page requise est chargée en mémoire, son entrée de table de pages est mise à jour pour refléter sa présence (bit valide défini), et le processus qui a provoqué le page fault peut reprendre son exécution.

- Le coût total d'un page fault est dominé par les opérations d'accès au disque, qui sont plusieurs ordres de grandeur plus lentes que les opérations en mémoire. Le temps total peut varier de quelques millisecondes à plusieurs secondes dans les cas extrêmes (par exemple, si le disque est très sollicité ou si une grande quantité de pages doit être échangée). C'est pour cette raison que les systèmes d'exploitation tentent de minimiser les page faults grâce à des stratégies de gestion de mémoire efficaces et à l'utilisation de techniques telles que le pré-chargement de pages et le réglage des algorithmes de remplacement de pages.

- Algorithme de page replacement pour économiser ce cout

- L'anomalie de Belady est un phénomène contre-intuitif lié aux algorithmes de remplacement de pages dans les systèmes de gestion de mémoire virtuelle. Elle démontre qu'augmenter le nombre de cadres de pages (slots de mémoire physique alloués pour stocker les pages de mémoire virtuelle) peut paradoxalement entraîner une augmentation du nombre de page faults (page faults), plutôt qu'une diminution comme on pourrait s'y attendre.

- Des algorithmes comme LRU (Least Recently Used) ou ceux basés sur des heuristiques plus sophistiquées ont été développés pour éviter de telles anomalies et améliorer l'efficacité de la gestion de la mémoire. (picture of question)

FIFO (First-In, First-Out)

FIFO évince la page la plus ancienne en mémoire, indépendamment de son taux d'utilisation. Cet algorithme suppose qu'une page qui est en mémoire depuis longtemps est probablement moins nécessaire que les pages plus récentes. C'est le plus simple des algorithmes de remplacement de pages, mais il peut conduire à des performances sous-optimales en raison de l'anomalie de Belady, où ajouter plus de cadres de pages peut parfois augmenter le nombre de page faults.

Avantages:

- **Simplicité**: FIFO est très simple à comprendre et à implémenter.
- **Juste**: Chaque page a la même durée de vie en mémoire.

Inconvénients:

- **Anomalie de Belady**: Comme mentionné, FIFO peut subir l'anomalie de Belady, où augmenter le nombre de cadres de page augmente le nombre de page faults.
- **Pas de distinction entre les pages**: Ne tient pas compte de la fréquence ou de la récence d'accès des pages.

Optimal (OPT)

Optimal est un algorithme théorique qui évince la page qui ne sera pas utilisée pendant la plus longue période future. Il fournit le taux de page fault le plus bas possible, servant de benchmark

pour évaluer les performances d'autres algorithmes. Cependant, sa mise en œuvre pratique est irréalisable car elle nécessite la connaissance des demandes de pages futures.

Avantages:

- **Performance**: Fournit le nombre minimal de page faults. Il est donc considéré comme l'algorithme optimal.

Inconvénients:

- **Non réalisable en pratique**: Requiert une connaissance préalable des pages qui seront accédées, ce qui n'est pas possible dans la plupart des cas réels.

LRU (Least Recently Used)

LRU supprime la page qui n'a pas été utilisée pendant la période la plus longue, s'appuyant sur l'hypothèse que les pages récemment utilisées sont plus susceptibles d'être utilisées à nouveau. LRU est plus efficace que FIFO dans de nombreux scénarios car il considère le modèle d'accès des pages. Cependant, sa mise en œuvre peut être coûteuse car elle nécessite de suivre l'ordre d'accès à chaque page.

Avantages:

- **Efficacité**: Réduit le nombre de page faults en supposant que les pages les plus récemment utilisées seront probablement utilisées à nouveau.
- **Pragmatique**: Plus réaliste et généralement performant dans les cas d'usage réels.

Inconvénients:

- **Complexité de mise en œuvre**: Peut être difficile et coûteux à implémenter précisément, notamment en ce qui concerne le suivi de l'ordre d'accès aux pages.

Bit de Second Chance (Clock)

Le **Bit** de Second Chance améliore FIFO en donnant une "seconde chance" aux pages avant de les évincer. Lorsqu'une page atteint le début de la file FIFO et est candidate à l'éviction, si son bit de référence est défini (indiquant qu'elle a été accédée récemment), ce bit est effacé et la page est replacée à la fin de la file, évitant son éviction immédiate. Cela permet de conserver des pages actives en mémoire un peu plus longtemps.

Avantages:

- **Compromis efficace**: Améliore FIFO en offrant une seconde chance aux pages, ce qui permet d'éviter de remplacer des pages fréquemment accédées.
- **Implémentation simple**: Plus facile à implémenter que LRU tout en offrant de meilleures performances que FIFO pur.

Inconvénients:

- **Approximation de LRU**: N'est qu'une approximation de LRU et peut ne pas être aussi efficace dans tous les cas.

Bit de Référence

Le **Bit de Référence** est utilisé dans divers algorithmes de remplacement pour suivre si une page a été accédée (lue ou écrite) pendant un certain intervalle de temps. Dans certains algorithmes, comme le Clock, les bits de référence aident à décider quelle page évincer. Une page avec un bit de référence à zéro est considérée comme candidate à l'éviction, tandis qu'un bit à un indique une page récemment utilisée. Les bits de référence fournissent une façon simplifiée de mesurer l'activité des pages pour améliorer les décisions de remplacement.

Avantages:

- **Flexibilité**: Permet des stratégies sophistiquées pour détecter les pages fréquemment utilisées.
- **Amélioration de FIFO et LRU**: Peut être utilisé pour améliorer les algorithmes FIFO et LRU en ajoutant une mesure de la fréquence d'accès.

Inconvénients:

- **Complexité**: Nécessite un mécanisme supplémentaire pour régulièrement réinitialiser les bits de référence, ce qui peut introduire une surcharge.
- **Nécessité de compromis**: La façon dont les bits de référence sont utilisés doit équilibrer précision et performance, nécessitant souvent un compromis.