M2-SEM

Concepts avancés d'architecture

Cours 3.1

Gestion Mémoire Virtuelle

Année 2020-2021

Pr. R. BOUDOUR

Terminologie

- **Multi-programmation** (plusieurs tâches en mémoire) → organisation
- **Mémoire contiguë :** Ensemble du code objet stocké en un seul bloc dans un espace mémoire donné
- **Mémoire non contiguë :** Le code objet est réparti en blocs séparés (surtout utilisé en mémoire virtuelle)
- **Technique non préemptive :** Une tâche reste en mémoire jusqu'à sa terminaison
- **Technique préemptive** : Le système peut redistribuer l'espace mémoire à une ou plusieurs autres tâches
- **Partition fixe** : La mémoire est partagée en blocs de taille définie
- **Partition variable** : La taille des blocs est ajustée selon les besoins

- **Page** : unité mémoire transférée entre disque et mémoire principale.
- **Défaut de page :** Quand un programme accède à un emplacement mémoire virtuelle absent de la mémoire principale
- **Translation d'adresse :** Le processus de recherche d'une adresse physique qui correspond à une adresse virtuelle

Terminologie

- **Adresse logique ou adresse virtuelle (*virtual address)***

  Adresse générée par la CPU.
- **Adresse physique**

  Adresse vue par l'unité de mémoire.

Terminologie

- **Mémoire :** grand tableau de mots (octets), chacun possédant sa propre adresse.
- **La CPU (processeur)** extrait les instructions de la mémoire en fonction de la valeur d'un compteur d'instructions.
- **Système de gestion de la mémoire** (*Memory manager): partie du SE qui gère la* hiérarchie de stockage
  - Suivre les parties de la mémoire qui sont utilisées ou non utilisées.
  - Allouer/libérer espace mémoire aux processus.
  - Contrôler le *swapping entre la mémoire principale et le disque*.

Introduction

- La mémoire est une ressource importante qui doit être gérée avec attention.
- Même si la quantité de mémoire d'un ordinateur a beaucoup augmenté, la taille des programmes s'est accrue aussi.
- La situation idéale serait de donner à chaque programmeur une mémoire :
  - *infiniment grande,*
  - *infiniment rapide,*
  - *non volatile*
  - *et, de plus, bon marché.*

  La technologie ne fournit pas de telles mémoires.
- **Hiérarchisation de la mémoire: les ordinateurs ont une petite quantité de** mémoire très rapide, chère et volatile (mémoire RAM) et beaucoup de gigabytes de mémoire plus lente, bon marché et non volatile.
- Le SE a le rôle de coordonner l'utilisation des différentes mémoires

Introduction

- Jusqu'à présent nous avons fait l'hypothèse que le programme machine était intégralement présent dans la mémoire centrale. Ceci impose donc que la taille du programme soit plus petite que la taille de la mémoire physique.

- Cette hypothèse implique que le compilateur attribue à chaque donnée ou instruction une adresse physique en mémoire centrale. Il n'est pas toujours possible de tenir cette hypothèse, ni même souhaitable en terme d'efficacité.

- Le processeur est ralenti par les autres modules de l'ordinateur (accès mémoire centrale, entrées-sorties,...), et n'est donc pas toujours utilisé au mieux.

- Une manière d'améliorer les performances globales est de permettre à plusieurs utilisateurs (programmes machines) placés simultanément en mémoire de se partager le processeur.

- Pour utiliser le processeur les instructions machine des programmes doivent être en mémoire centrale mais la totalité des instructions de tous les programmes peuvent ne pas être simultanément en mémoire centrale.

- Ainsi l'intégralité de tous les programmes peut ne pas être présente en mémoire en même temps.

Introduction

→ partager le processeur entre plusieurs programmes

=

partager la mémoire centrale entre des parties de programmes présentes

- Dans ce cas ni le compilateur ni le chargeur (charger l'exécutable en MC) ne peuvent a priori attribuer aux données et aux instructions une adresse physique en mémoire centrale.

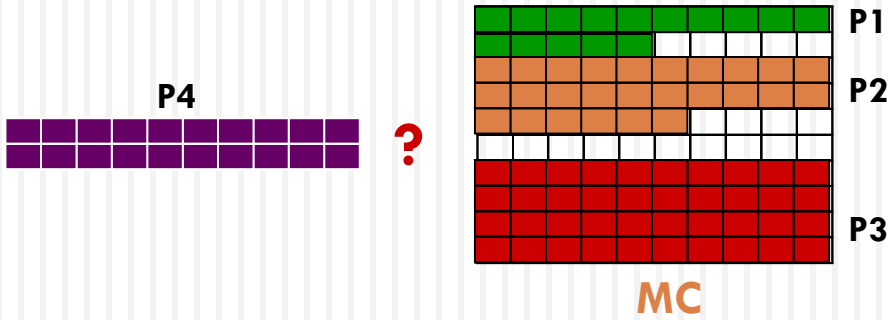
- On est amené à séparer l'espace d'adressage du programme et l'espace d'adressage physique en définissant des adresses virtuelles pour les instructions et les données.

- Cette adresse virtuelle ne dépend que de l'espace d'adressage du programme et ne tient pas compte de l'adressage physique (réel) de la mémoire centrale.

- C'est seulement au moment de l'exécution d'une instruction que la correspondance « adresse virtuelle », « adresse physique » est établie.

En d'autres termes

- **Avant la mémoire virtuelle : exemple**
  - Mémoire centrale est coûteuse, et de taille limitée.
  - Programmes gourmands en mémoire et qui ne "tiennent pas" toujours en MC.
  - On veut Augmenter le nombre de processus en MC ?



Trouver un emplacement pour le processus P4 ?

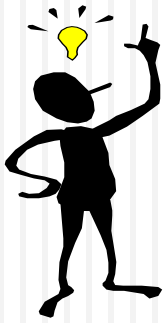
*Il faut bien **gérer la Mémoire** centrale pour éviter ce genre de situations.*

En d'autres termes

- **Avec mémoire virtuelle**
  - Le principe de mémoire virtuelle a été inventé à la fin des années 1950 pour pallier à ce problème.
  - Les ordinateurs travaillent avec des adresses virtuelles ne correspondant pas à l'espace physique.
    - Les données peuvent être ainsi dans la RAM mais aussi sur le disque dur de l'ordinateur,
    - voire dans la RAM ou sur le disque dur d'une autre machine sur le réseau local.
    - La RAM est donc utilisée seulement pour stocker les données les plus utilisées.

En d'autres termes

- **Objectif** : Offrir à l 'utilisateur (aux utilisateurs) un espace d 'adressage plus grand que la capacité réelle de la mémoire (physique) sans pour autant augmenter (de beaucoup) le temps d 'accès à la MC.
  - Offrir la capacité du disque dur avec le temps d'accès de la DRAM.
  - **Exemple** : la MC (espace physique) = 64 Ko,
    Si 4 utilisateurs de 64Ko chacun = total 256 Ko : Impossible

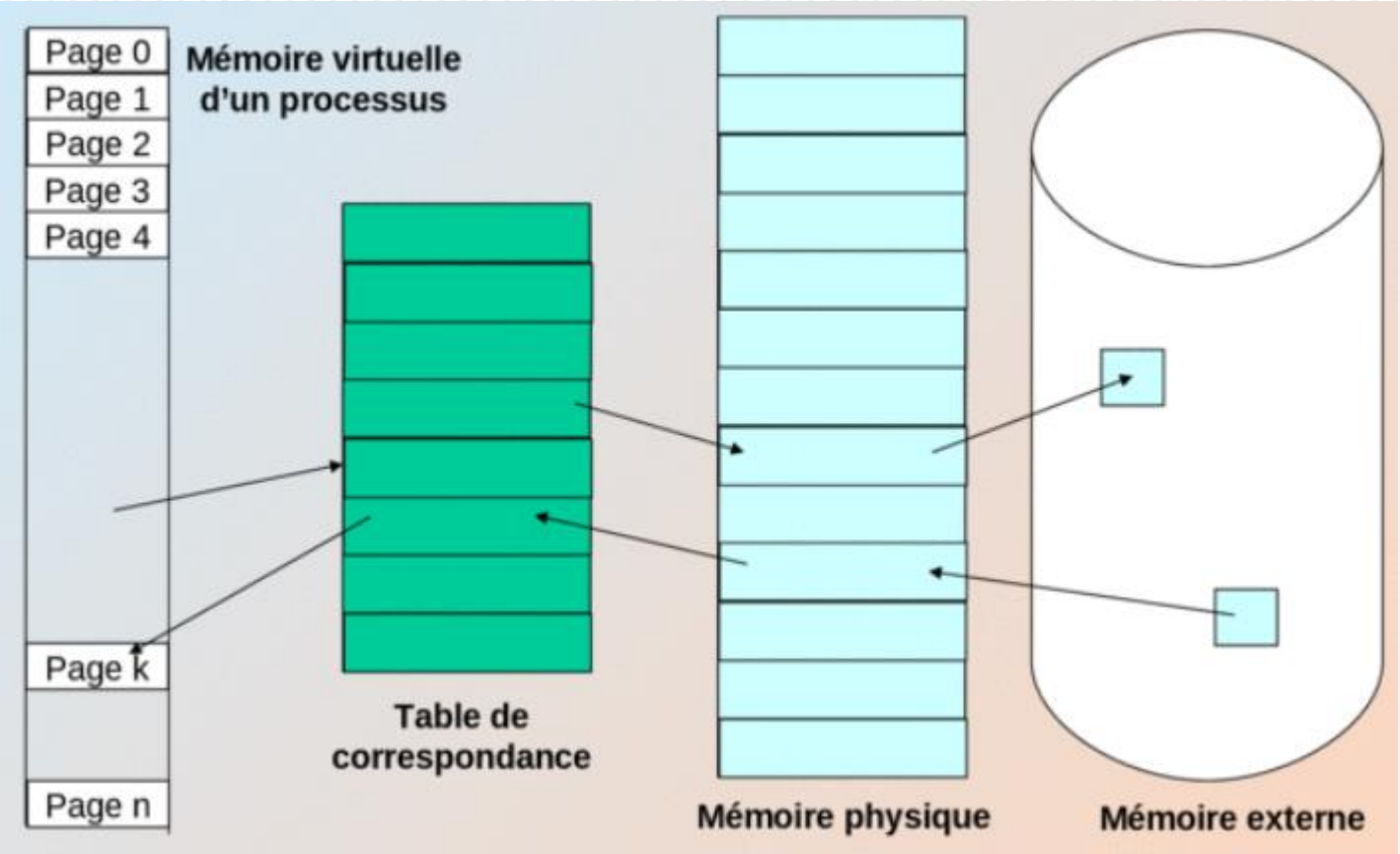


- **Solution** :
  - **partager l'espace entre tous les utilisateurs en le divisant en morceaux (de tailles fixes ou de tailles variables)**
  - et mettre tout en mémoire secondaire (MS : de grande taille et de faible coût exemple disque dur)
  - Les morceaux (prog et données) sont chargés de la MS vers la MC à la demande (lorsqu'on en a besoin) (similaire au cache)

Définitions

- **Mémoire virtuelle –** technique permettant de faire exécuter des processus qui ne se trouvent pas en totalité en mémoire ; ils peuvent donc utiliser un espace d'adressage logique supérieur à l'espace physique du système.
- **Cette technique sépare la mémoire logique de l'utilisateur de la mémoire physique du système**
  - Simplifie le partage des espaces d'adressage.
  - Simplifie la création des processus.
- Deux techniques d'implémentation :
  - ***Pages à la demande*** : la technique la plus courante.
  - ***Segment à la demande*** : plus compliquée et rarement utilisée (IBM OS2, Burroughs)
- Espace d'adressage virtuel : espace logique du processus, vu comme un plage *continue* d'adresses commençant à 0

Organisation de la MV



Mémoire virtuelle

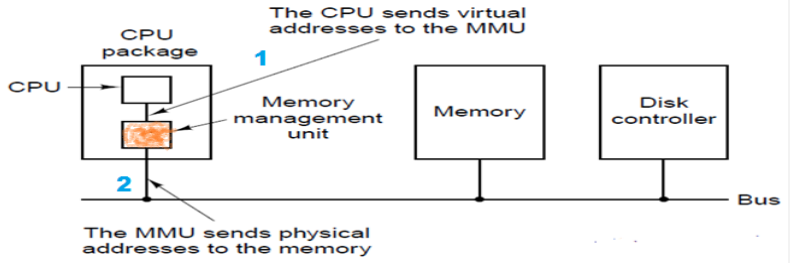
- En général, la mémoire virtuelle et la mémoire physique sont structurées en unités d'allocations :
  - **Pagination**
    - pages pour mémoire virtuelle
    - cadres ou frames pour la mémoire physique
    - taille d'une page est égale à celle d'un cadre
  - **Segmentation**
    - Entités sont des segments
- Lorsqu'un processus est en cours d'exécution, seule une partie de son espace d'adressage est en mémoire

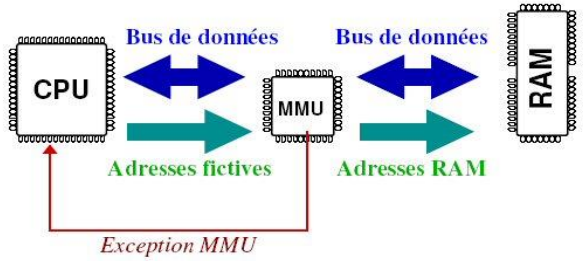
La mémoire virtuelle permet :

- d'augmenter le taux de multiprogrammation
- de mettre en place des mécanismes de protection de la mémoire
- de partager la mémoire entre processus

MMU

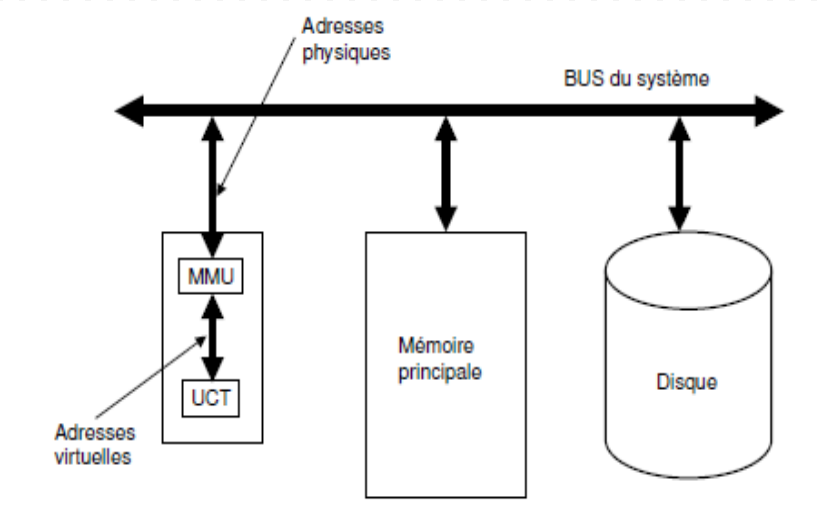
- **MMU (*Memory Management Unit*) :**
  Dispositif matériel qui fait la conversion des adresses virtuelles en adresses physiques
- Le programme utilisateur ne perçoit jamais les adresses physiques ; il traite des adresses logiques.
- *En pratique, la MMU peut être une puce externe au processeur située entre le processeur et la RAM ou être intégrée directement dans la puce qui contient le processeur.*





MMU

- La MMU a été intégrée aux microprocesseurs, à partir du 80386 pour la gamme Intel x86, à partir du 68030 pour la gamme Motorola 680x0.
- L'unité de gestion mémoire fait désormais partie intégrante de tous les microprocesseurs récents.



MMU : Fonctionnement

- Le système programme la MMU en déclarant une zone mémoire précise comme appartenant à un programme précis (une zone exécutable de la mémoire).
  - L'utilisation de translation d'adresse est souvent utilisée conjointement à la protection mémoire afin que les variables du programme commencent à l'adresse 0. Si une tentative d'accès à la mémoire hors plage est détectée, une interruption est levée par la MMU.
  - Celle-ci est interceptée par le processeur et cela a généralement pour effet de stopper le programme, qui reçoit par exemple : un signal *de violation de segmentation*

MMU : Fonctionnement

- Les adresses virtuelles, générées par les compilateurs et les éditeurs de liens, sont des couples composés d'un numéro de page et d'un déplacement relatif au début de la page.
  - Les adresses virtuelles référencées par l'instruction en cours d'exécution doivent être converties en adresses physiques.
  - Cette conversion d'adresse est effectuée par le MMU, qui sont des circuits matériels de gestion.
  - Nous allons considérer la page : l'unité de transfert. Le MMU vérifie si l'adresse virtuelle reçue correspond à une adresse en mémoire physique. Si c'est le cas, le MMU transmet sur le bus de la mémoire l'adresse réelle, sinon il se produit un défaut de page.

MMU : Fonctionnement

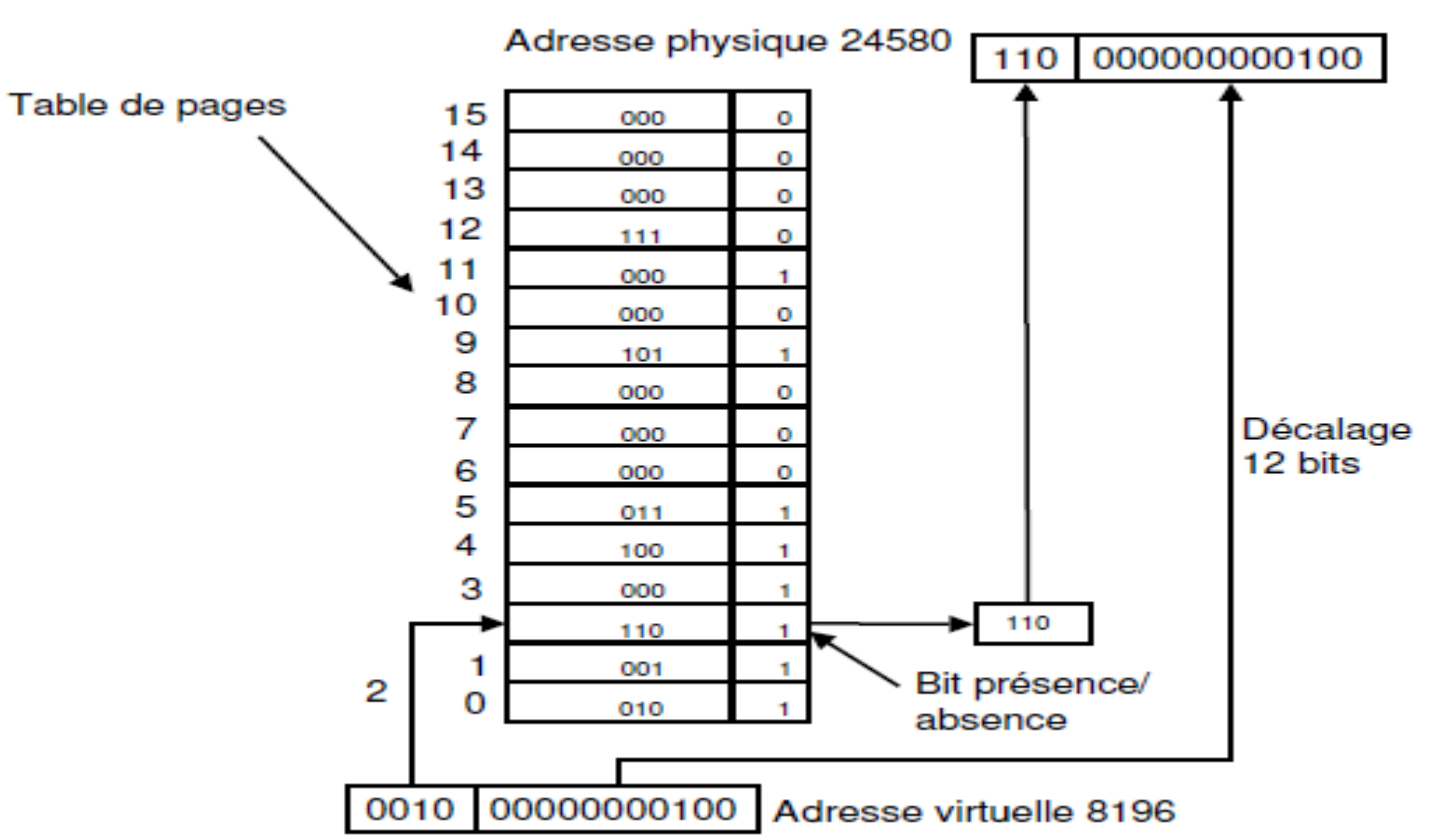
- Un défaut de page provoque un déroutement (ou TRAP) dont le rôle est de ramener à partir du disque la page manquante référencée.
- La correspondance entre les pages et les cases est mémorisée dans une table appelée **Table de pages** (**TP**).
- Le nombre d'entrées dans cette table est égal au nombre de pages virtuelles.
- La **Table de pages** d'un processus doit être en totalité ou en partie en mémoire centrale lors de l'exécution du processus. Elle est nécessaire pour la conversion des adresses virtuelles en adresses physiques

Exemple

- La MMU reçoit, en entrée une adresse virtuelle et envoie en sortie l'adresse physique ou provoque un déroutement.
- Dans l'exemple ci-dessous, l'adresse virtuelle est codée sur 16 bits. Les 4 bits de poids fort indiquent le numéro de page, comprise entre 0 et 15. Les autres bits donnent le déplacement dans la page, entre 0 et 4095.
- Le MMU examine l'entrée dans la **Table de pages** correspondant au numéro de page, dans ce cas 2.
  - Si le bit de présence est à 0, la page n'est pas en mémoire alors le MMU provoque un déroutement.
  - Sinon, il détermine l'adresse physique en recopiant dans les 3 bits de poids le plus fort le numéro de case (110) correspondant au numéro de page (0010), et dans les 12 bits de poids le plus faible de l'adresse virtuelle.
  - L'adresse virtuelle 8196 (0010 0000 0000 0100) est convertie alors en adresse physique 24580 (1100 0000 000 0100) comme le montre la figure.

Exemple

- Opérations MMU



Construction d'une adresse

- Un programme est décomposés en blocs.

- Dans l'espace d'adressage du programme nous appelons les blocs, des blocs logiques.

- Quand un bloc logique est présent en mémoire principale il correspond à un bloc physique. Le bloc physique a la même taille que le bloc logique, il est implanté en mémoire centrale à l'adresse, « adresse_début_bloc ».

- Pour déterminer l'adresse effective (physique) d'un mot à partir de son adresse virtuelle, on dispose d'une table de correspondance (table_des_blocs) qui a **autant d'entrées qu'il y a de blocs logiques**.

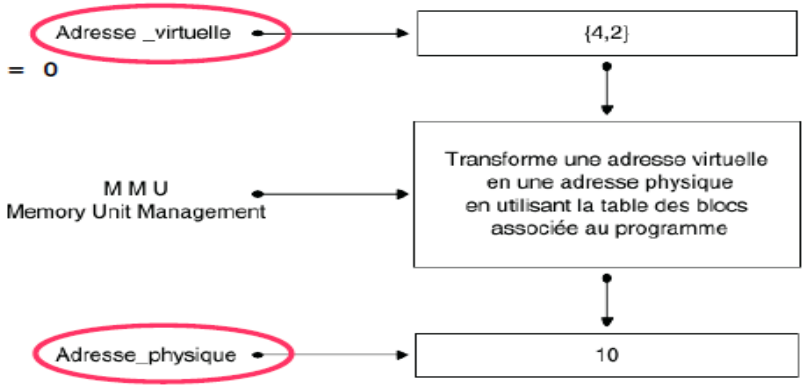
- Pour chaque entrée on trouve un **indicateur de présence** ou d'absence du bloc logique en mémoire et dans le cas de la présence du bloc physique, l'**adresse physique du début de ce bloc**, soit adresse_début_bloc. Dans ces conditions :
adresse _physique = table_des_blocs[numéro_de_bloc_logique, 1] + déplacement

Les blocs d'un programme ne sont pas tous nécessairement présents en mémoire centrale.

Correspondance AV-AP

- Pour établir la correspondance « adresse virtuelle », « adresse physique » le mécanisme de gestion de la mémoire virtuelle utilise un module matériel : le **MMU** (Memory Management Unit). Ce module reçoit en entrée une adresse virtuelle et convertit cette adresse en une adresse physique en utilisant la table **table_des_blocs** associée au programme en cours d'exécution.

```
début
si table_des_blocs [numéro_de_bloc_logique, 0] = 0
alors
     défaut de bloc physique;
     lire bloc_logique sur disque;
     si place libre en mémoire physique
     alors
          charger bloc_logique;
     sinon
          algorithme de remplacement d'un bloc;
          modifier la table table_des_blocs;
     finsi
finsi
calculer adresse_physique;
fin
```



- Lorsqu'un bloc logique n'est pas présent en mémoire (défaut de bloc physique) le MMU produit un déroutement vers le **système d'exploitation** (interruption logicielle) qui examine la place disponible en mémoire centrale. S'il n'y a pas de place pour charger le bloc logique nécessaire à la poursuite de l'exécution, le système d'exploitation exécute alors un algorithme de **remplacement de bloc**.

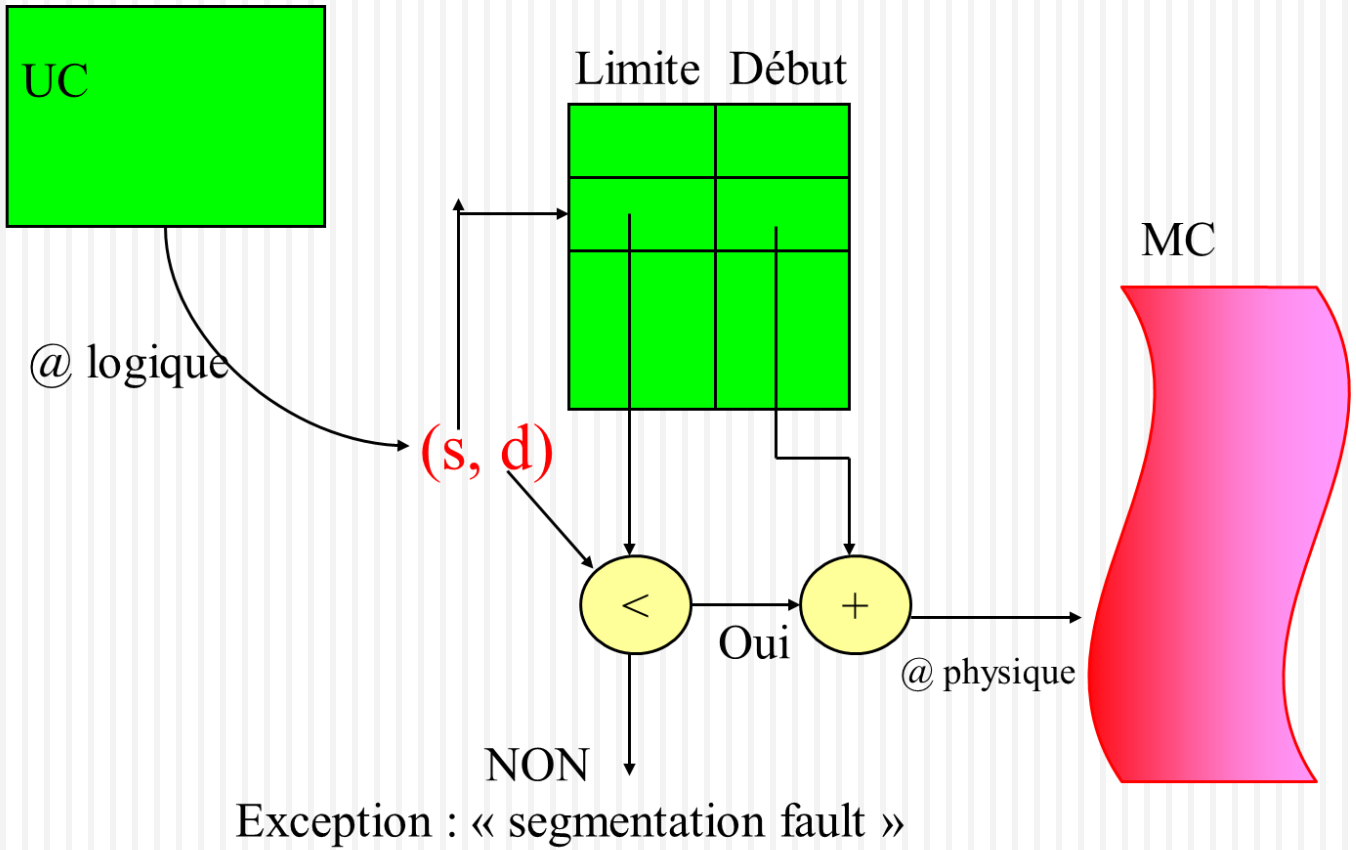
Implémentation de la MV

- La mémoire virtuelle (MV) est implémentée sous deux formes principales :
  - Segmentation
  - Pagination

MV : La segmentation

- **Segmentation :**
  - plusieurs espaces d 'adressage
  - Division logique : segment code, segment donnée, segment pile, ….
  - Visible du programmeur
  - les adresses dans le segment vont de 0 à adrMax (émiettement externe) $\Rightarrow$ ramasse miette (compacter la MC)
  - taille d'un segment varie en cours de l exécution (donnée, pile)
  - permet une meilleure protection : A chaque segment est associé des bits de contrôle d'accès
- Une adresse segmentée se compose : **numéro segment, déplacement dans le segment** (s,d)

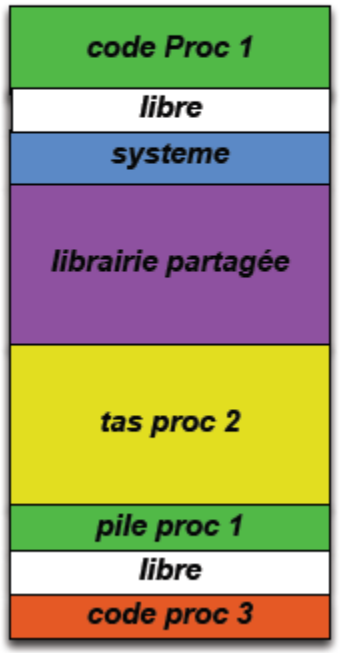
Segmentation pure



Segmentation

La mémoire segmentée :

adresse = numéro de segment + décalage



Les gains

Facilité de compilation (édition de liens).
Facilité de partage mémoire.
Facilité de protection.

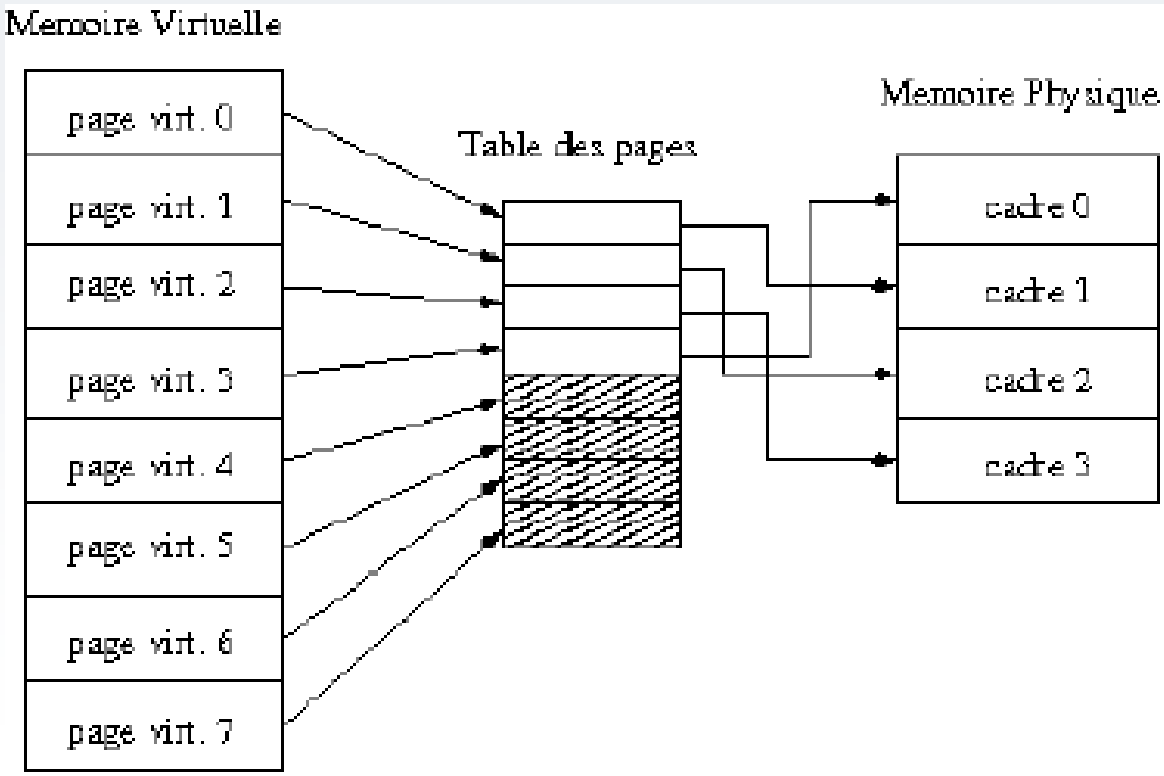
Les défauts

La fragmentation externe.
Ralentissement ?

MV : Pagination

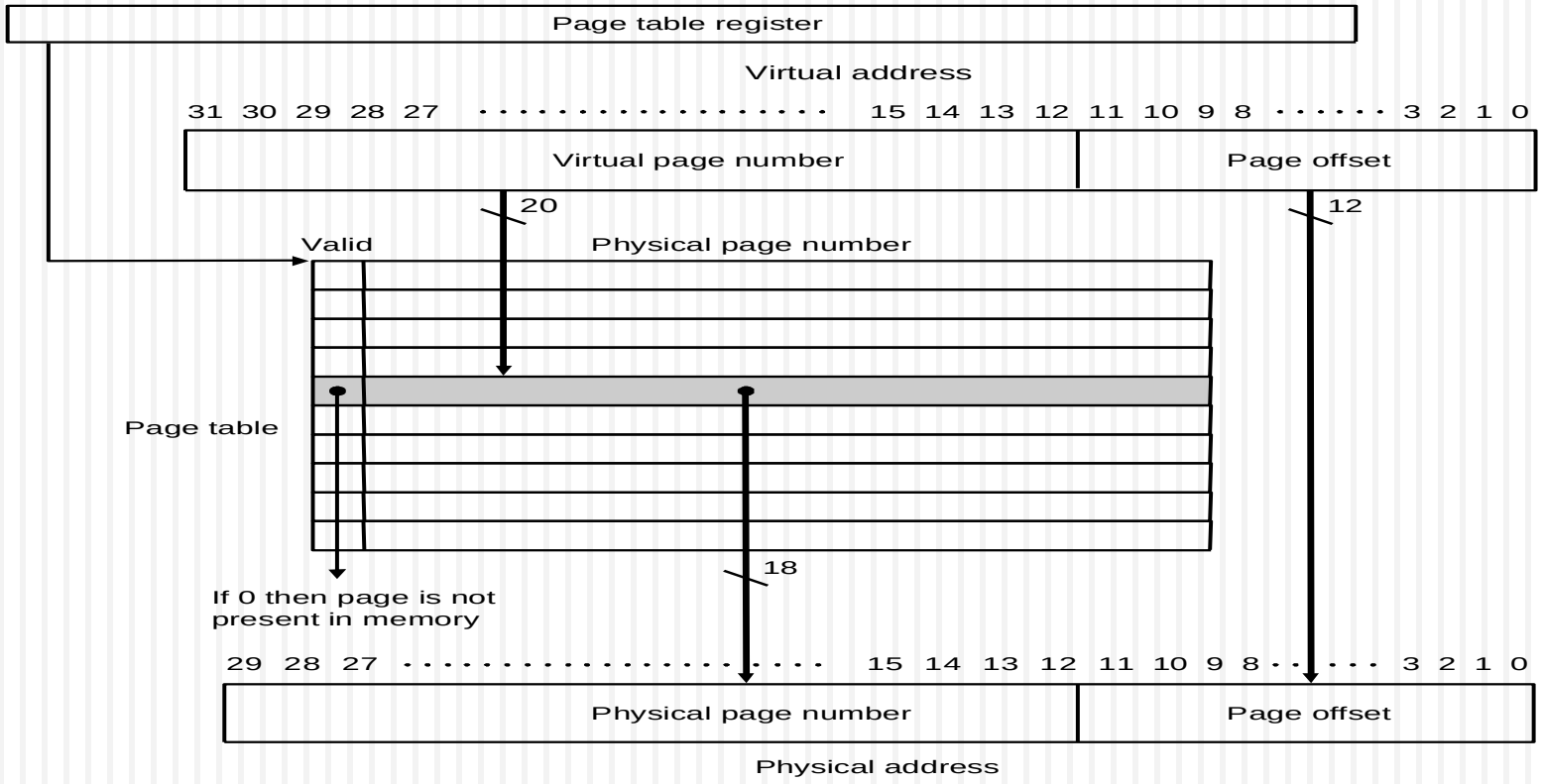
- L'espace mémoire de chaque programme est divisé en unités appelées pages.
- La MC est divisée en cadres. Une page a une correspondance à un cadre (*même taille*)
- *Les adresses dans l'espace virtuel sont appelées adresses virtuelles.*
- *A l'exécution, la MMU (Memory Management Unit) fait la correspondance entre les adresses virtuelles et physiques à l'aide d'une table de pages.*

MV : Pagination



MV : Pagination

- Le début de la Table des pages est pointé par un registre : « Page Table Register »



Structure de la table des pages

- En général, chaque entrée de la table des pages comporte plusieurs champs :
  - Le bit de présence
  - Le bit de référence
  - Les bits de protection
  - Le bit de modification
  - Le numéro de cadre correspondant à la page

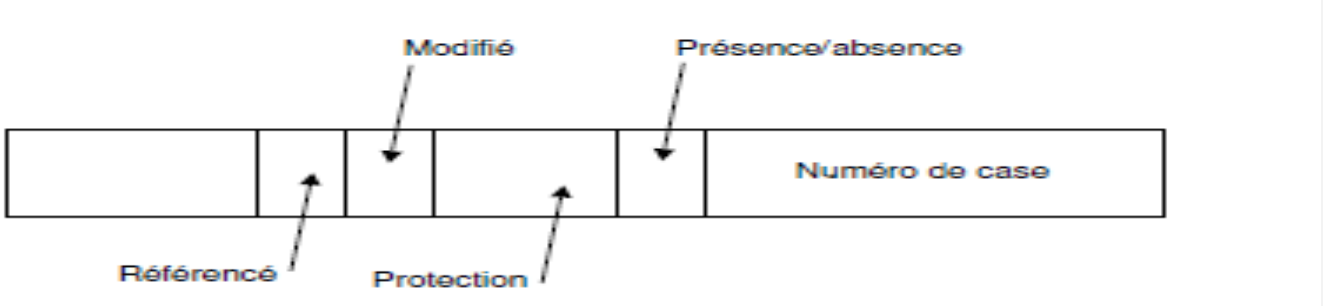
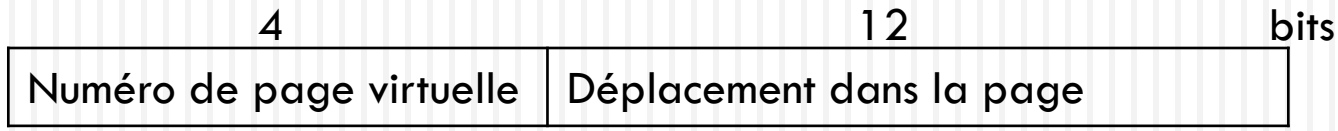


Table à un seul niveau

- Exemple :
  - **adresse virtuelle sur 16 bits**
    - 4 bits de poids forts indiquent le numéro de page
    - les autres bits donnent le déplacement dans la page : 0 à 4095



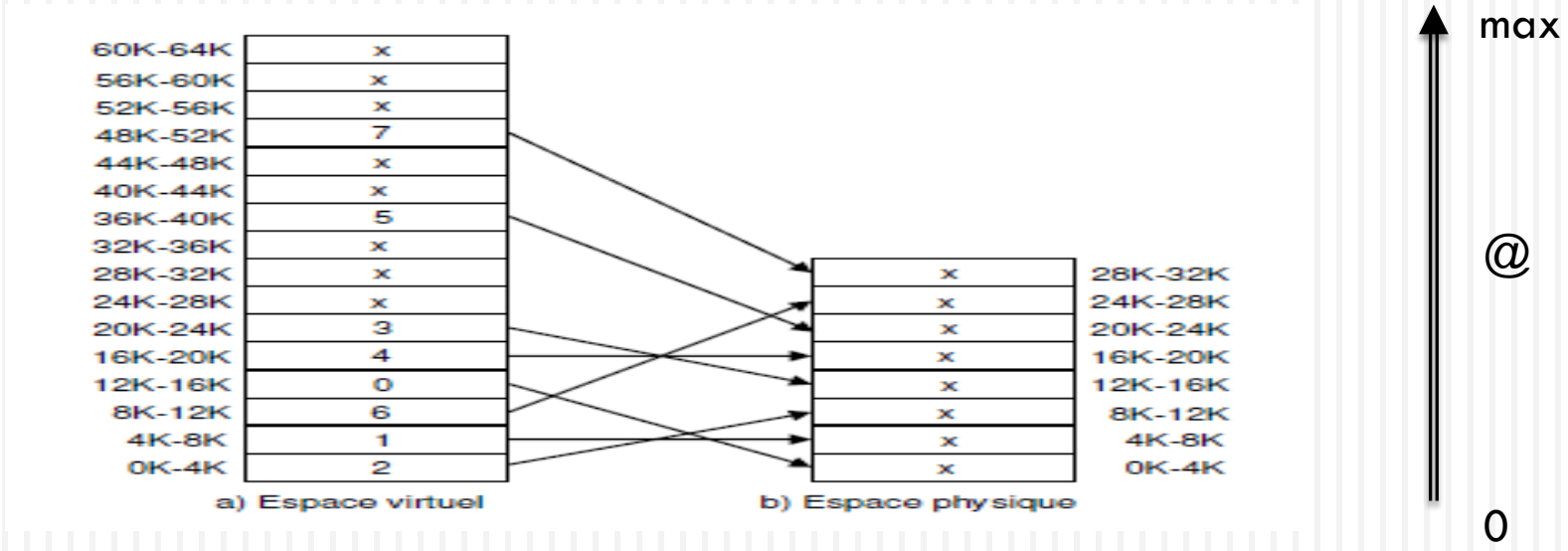


Table à un seul niveau

- Le MMU examine l'entrée dans la table des pages correspondant au numéro de page :
  - Si le bit de présence est à 0 : La page n'est pas en mémoire donc déroutement
  - Sinon, le MMU détermine l'adresse physique en recopiant dans les bits de poids fort le numéro de cadre (110) correspondant à (0010)
  - Ainsi, l'adresse virtuelle 8196 (0010 00000000 0100) est convertie en adresse physique 24580 (110 00000000 0100)

Numéro de page physique	Déplacement dans la page

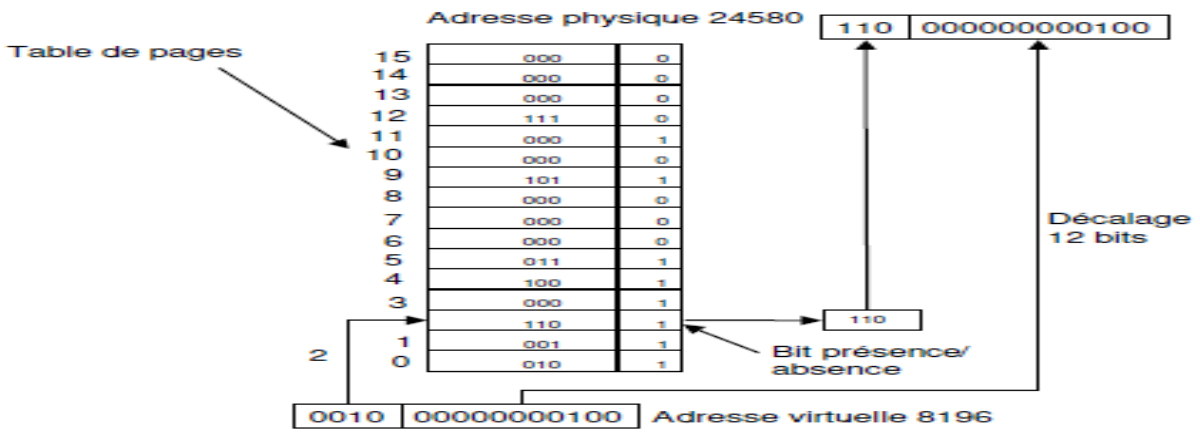


Table à un seul niveau

- **La table des pages occupe un grand espace**

NPV sur 20 bits	Déplacement sur 12 bits

- Il y a 2^{20} pages en MS, Taille de la page 2^{12} = 4 Ko
- La table des pages doit contenir 2^{20} entrées
- si NPP est sur 8 bits + @MS sur 20 bits + qq bits = 32 bits = 4 octets
  TP occupe : 4 * 2^{20} octets = 4 Mo en MC

Solution : utilisation de plusieurs niveaux de tables de pages et stocker seulement quelques tables de pages en MC

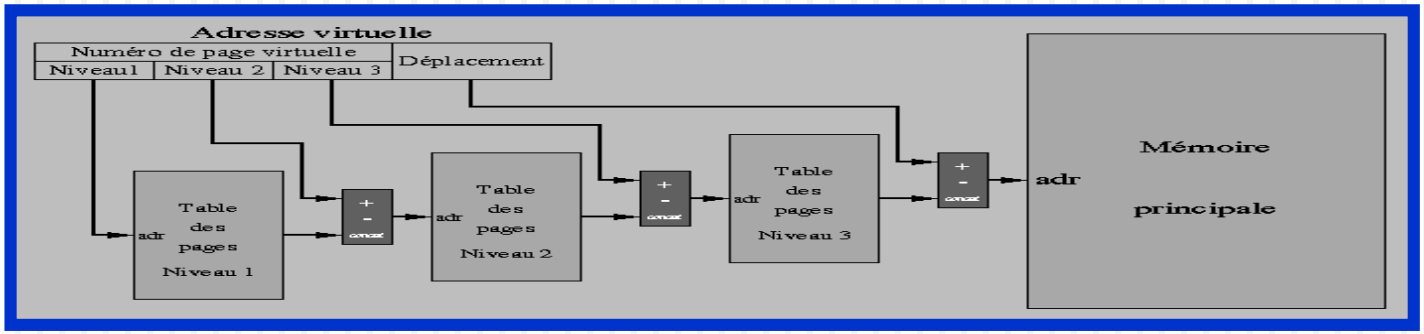
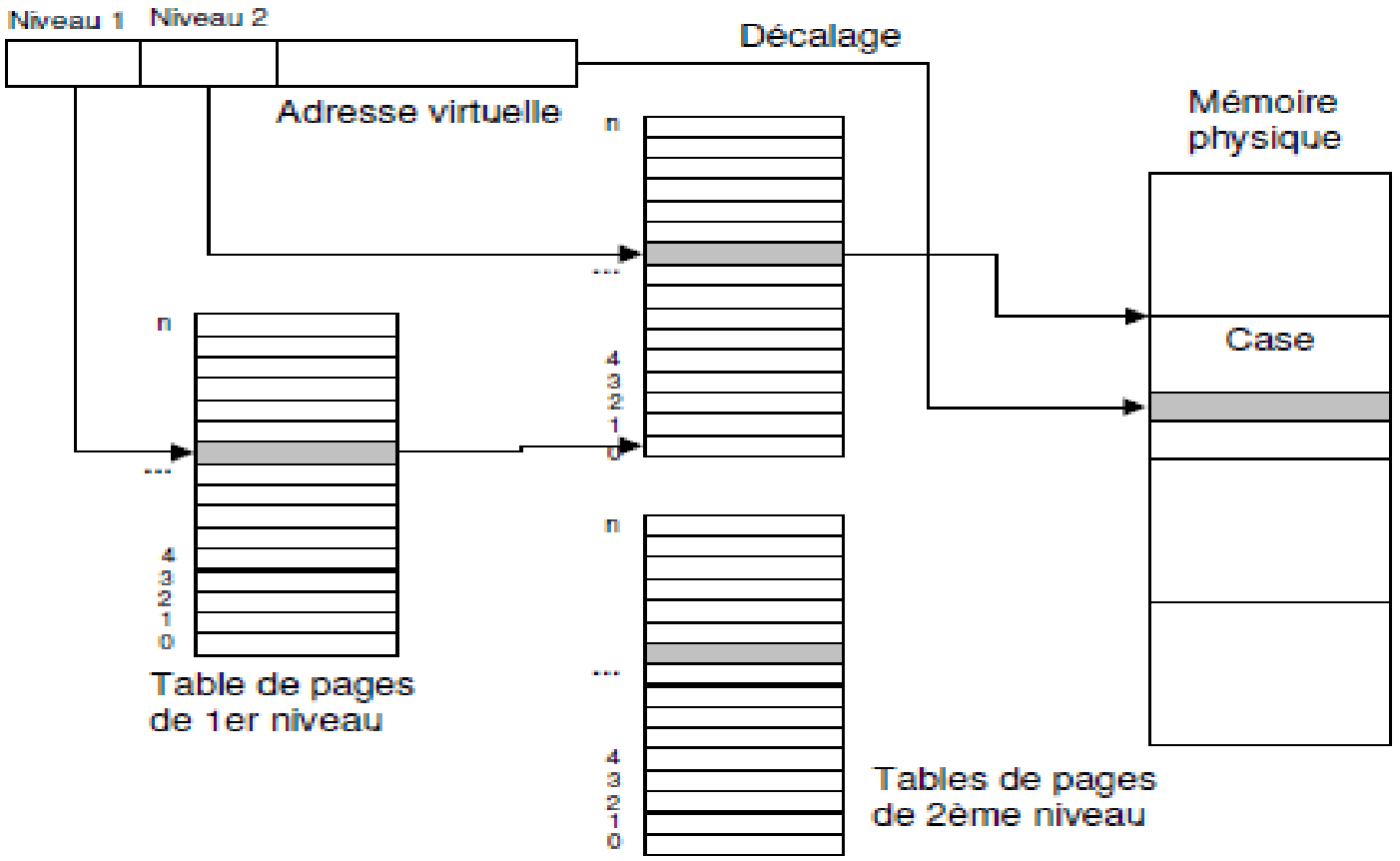


Table à plusieurs niveaux

- Pour éviter d'avoir des tables trop grandes en mémoire,
  - nombreux ordinateurs utilisent des tables de pages à plusieurs niveaux
  - charger uniquement quelques tables
- Exemple : table à deux niveaux (Cas Intel)
  - adresse : 32 bits
    - 1 table de 1er niveau
    - 1024 tables de 2ème niveau
  - une adresse virtuelle sera composée de trois champs
    - un pointeur sur la table de 1er niveau
    - un pointeur sur une table de 2ème niveau
    - un déplacement dans la page

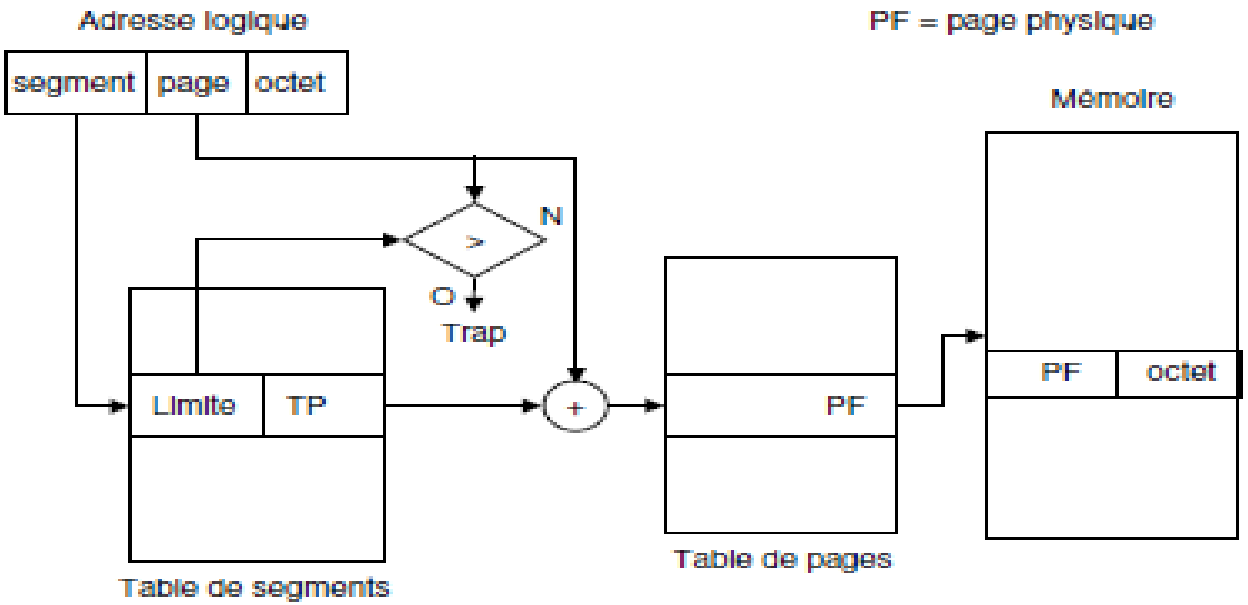
Table à deux niveaux



Segmentation paginée x86

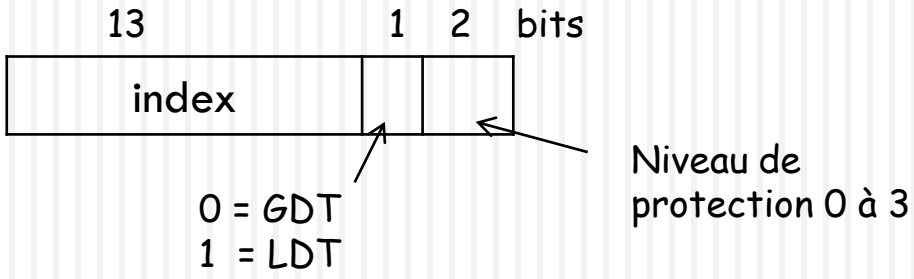
- Le processeur 80x86 utilise la segmentation paginée pour gérer la mémoire :
  - La mémoire physique est divisée en pages de 4 Ko
  - Le nombre maximum de segments par processus est de 16K
  - La taille d'un segment peut aller jusqu'à 4 Go
  - La conversion d'adresse :
    - adresse logique vers adresse linéaire
    - adresse linéaire vers adresse physique

Segmentation paginée



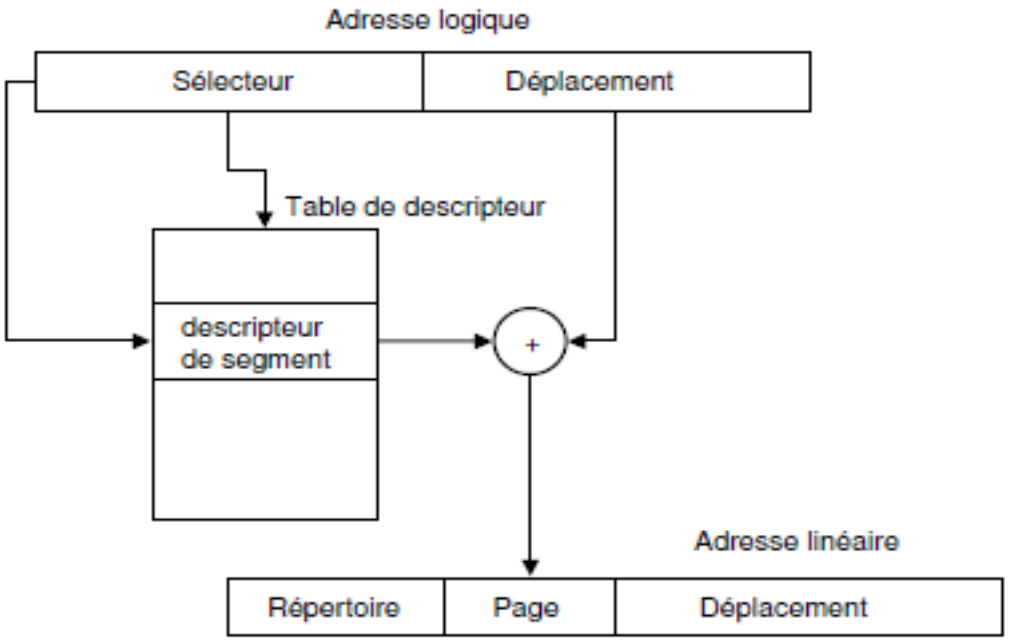
- **Sélecteur**

- Ex : 72_{10} = 0000000001001 0 00_2
  correspond à l'entrée 9
  de GDT : GDT + 9



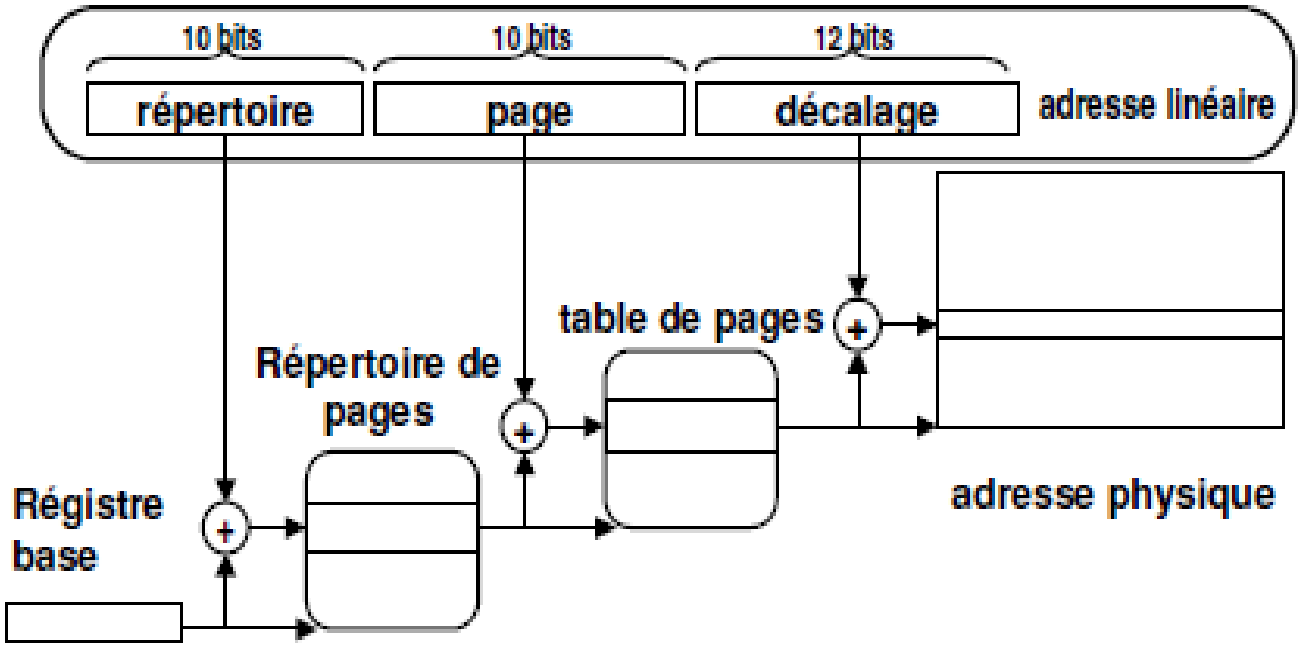
Segmentation paginée x86

- Translation d'adresse logique vers adresse linéaire



Segmentation paginée x86

- Adresse linéaire vers adresse physique



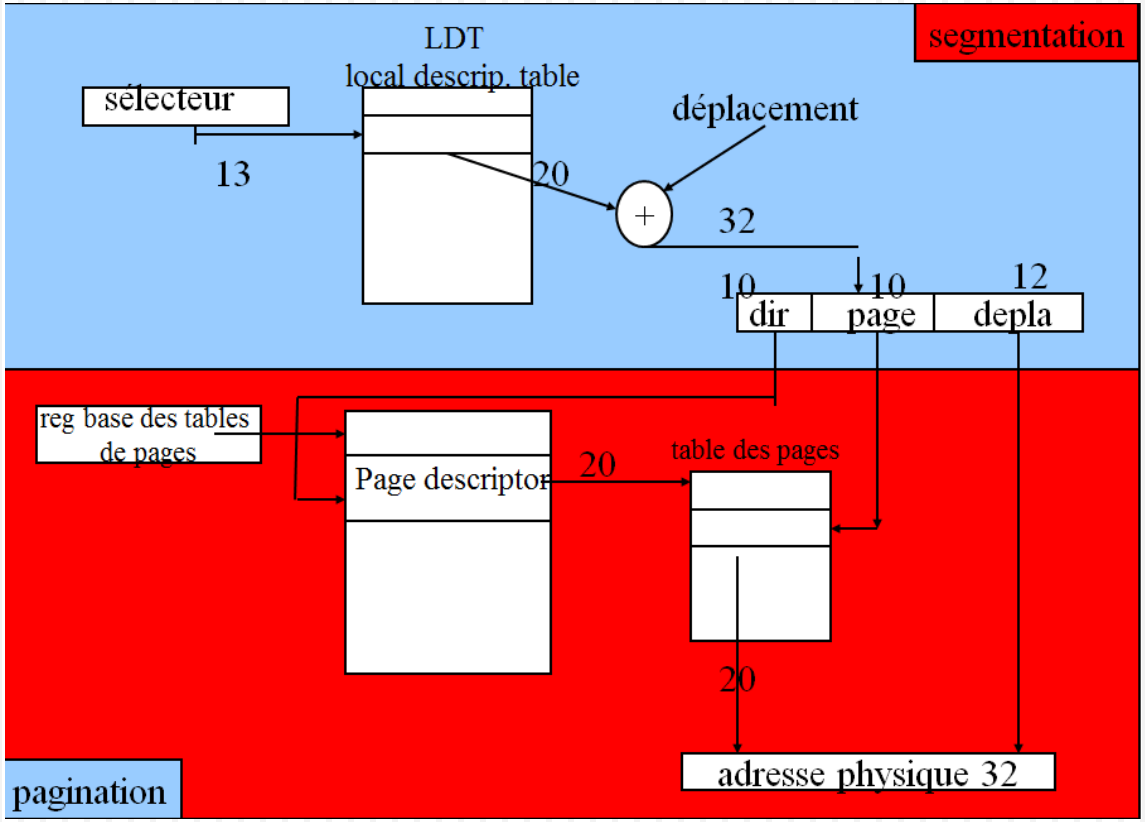
Segmentation paginée x86

- Une adresse linéaire (32 bits) se décompose de la façon suivante pour la partie pagination de la MMU (80x86) :

10 bits	10 bits	12 bits
DIR	PAGE	OFFSET

- **bits 22-31 :** L'index (DIR) dans la table de premier niveau, où est stockée l'adresse de début de la table de second niveau en RAM ;
- **bits 12-21 :** L'index dans la table (PAGE) de second niveau, où est stockée l'adresse de début de la page en RAM ;
- **bits 0-11 :** Le déplacement (OFFSET) dans la page

Segmentation paginée x86



Comparaison

	Pagination	Segmentation
Le programme a-t-il besoin de savoir que cette technique est utilisée ?	Non	Oui
Combien d'espace d'adressage linéaire y a-t-il ?	1	Beaucoup
L'espace d'adressage total peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Peut-on distinguer données et procédures et les protéger séparément ?	Non	Oui
Peut-on gérer facilement les tables dont la taille fluctue ?	Non	Oui
Pourquoi l'a-t-on inventée ?	Pour avoir un grand espace d'adressage	Pour séparer logiquement certains espaces d'adressage

Différence entre la pagination et la segmentation

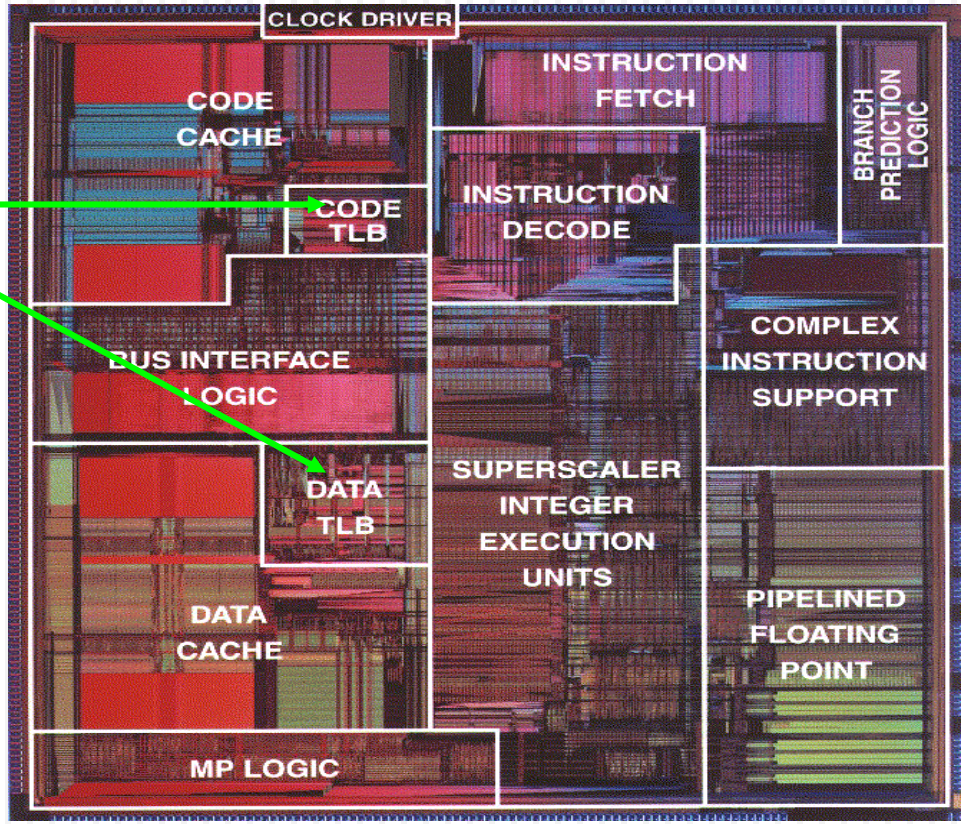
	Pagination	Segmentation
Définition	Une page a une taille de bloc fixe.	Un segment est de taille variable.
Fragmentation	La pagination peut entraîner une fragmentation interne.	La segmentation peut conduire à une fragmentation externe.
Adresse	L'adresse spécifiée par l'utilisateur est divisée par le CPU en un numéro de page + un décalage.	L'utilisateur spécifie chaque adresse par deux quantités un numéro de segment et le décalage (limite de segment).
Taille	Le matériel décide la taille de page.	La taille du segment est spécifiée par l'utilisateur.
Table	La pagination implique une table de pages qui contient l'adresse de base de chaque page.	La segmentation implique la table de segments qui contient le numéro de segment et le décalage (longueur du segment).

Différences : 80386 - 68030

	80386	68030
Taille de l'espace d'adressage virtuel (en octets) ?	2^{46}	2^{33}
Est-il possible de séparer espaces instruction et données ?	Non	Oui
Peut-on avoir une pure segmentation ?	Oui	non
Peut-on avoir une pure pagination ?	Oui	Oui
Peut-on avoir pagination et segmentation ?	Oui	Non
Nombre de niveaux de tables de page ?	2	4
Taille d'une page ?	4k	256o-32Ko
Taille d'une table de pages	4k	variable
Chaque page dispose-t-elle de bits accès/modification ?	Oui	Oui
Peut-on avoir des guichets ?	Oui	Non
Nombre de niveaux de protection ?	4	2

Pentium IV

Partie du Chip pour la gestion de la mémoire Virtuelle (traduction rapide)



Conclusion

- Le mécanisme de gestion de mémoire virtuelle n'est pas conceptuellement différent du mécanisme de gestion des caches.

- La différence porte essentiellement sur l'implantation de ces mécanismes. Par exemple dans le cas des caches, les algorithmes de remplacement sont pris en charge par le matériel alors que pour la mémoire virtuelle cette gestion est logicielle (système d'exploitation).

- Il y a plusieurs mécanismes possibles pour la gestion de la mémoire virtuelle, taille des blocs fixe ou variable, implication des programmeurs ou transparence au niveau de la programmation, implication de modules matériels ou non.


Thank's

QCM

- **Que permet le concept de mémoire virtuelle ?**
  - de gérer une partie de l'espace disque (mémoire secondaire) comme s'il s'agissait de mémoire principale.
  - d'exécuter des tâches qui ne peuvent physiquement tenir complètement en mémoire principale.
  - d'utiliser efficacement les caches mémoires L1 et L2.
  - d'augmenter la vitesse de commutation lors de la préemption des tâches.

QCM

- **Les techniques de "segmentation"**
  - sont similaires et ont les mêmes buts que ceux pour la "mémoire virtuelle
  - sont basées sur le découpage en modules, par le programmeur, du code et des données d'un processus
  - exploitent une "table de segments"
  - les 3 dernières réponses
  - aucune des 4 dernières réponses

QCM

- **Une "table des pages" fait correspondre à chaque page virtuelle des informations comme**
  - son emplacement en Mémoire Principale (s'il existe)
  - un bit pour savoir si le contenu de cette page réelle a été modifiée
  - des bits pour spécifier le degré de protection de la page en lecture/écriture/exécution
  - les 3 dernières réponses
  - aucune des 4 dernières réponses

QCM

- **La même adresse virtuelle utilisée par deux processus différents (plusieurs choix possibles)**
  - peut avoir la même traduction en adresse physique
  - est traduite par la même table de pages
  - n'existe pas : les adresses sont uniques
  - est traduite par deux tables de pages différentes
  - n'a pas toujours la même traduction en adresse physique
  - Aucune de ces réponses n'est correcte.

QCM

- **La MMU (Memory Management Unit) (1 choix possible) :**
  - est le code du système d'exploitation chargé de traduire chaque adresse virtuelle, manipulée par le programme s'exécutant sur le processeur, en adresse physique, manipulée par la mémoire
  - vérifie et corrige les modifications des bits de mémoire par les rayons cosmiques (mémoire ECC)
  - contrôle la température des barrettes de RAM
  - est un matériel qui traduit les adresses virtuelles manipulées par le programme s'exécutant sur le processeur en adresses physiques manipulées par la mémoire

QCM

- **Une table des pages (plusieurs choix possibles) :**
  - décrit la traduction des adresses virtuelles de tous les processus
  - décrit la traduction des adresses virtuelles d'un seul processus
  - est une structure de données creuse stockant la partie haute d'adresses physiques
  - est stockée dans la mémoire
  - est stockée dans la MMU
  - n'est jamais modifiée
  - est une structure de donnée creuse indexée par la partie haute des adresses virtuelles
  - est cachée dans le TLB
  - est modifiée par la table des pages
  - est une structure de donnée creuse stockant la partie haute des adresses virtuelles
  - est une structure de donnée creuse indexée par la partie haute des adresses physiques
  - est lue par la MMU
  - est modifiée par la MMU
  - Aucune de ces réponses n'est correcte.

QCM

- **Le TLB (Translation Lookaside Buffer)**
  - est la structure de donnée en mémoire permettant d'accélérer les traductions
  - est le cache des valeurs de la table des pages dans le processeur
  - Aucune de ces réponses n'st correcte