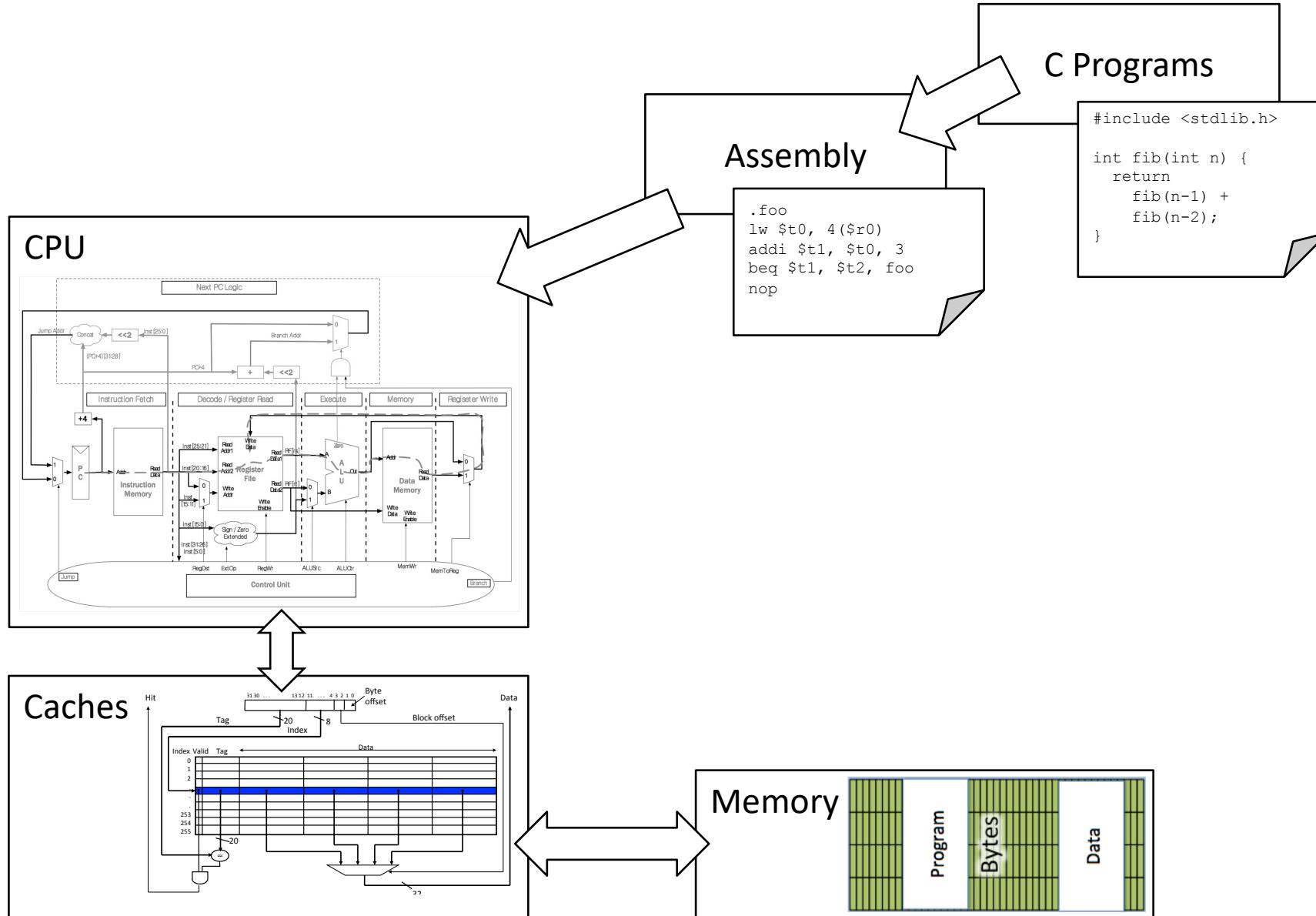


Architecture des Processseurs

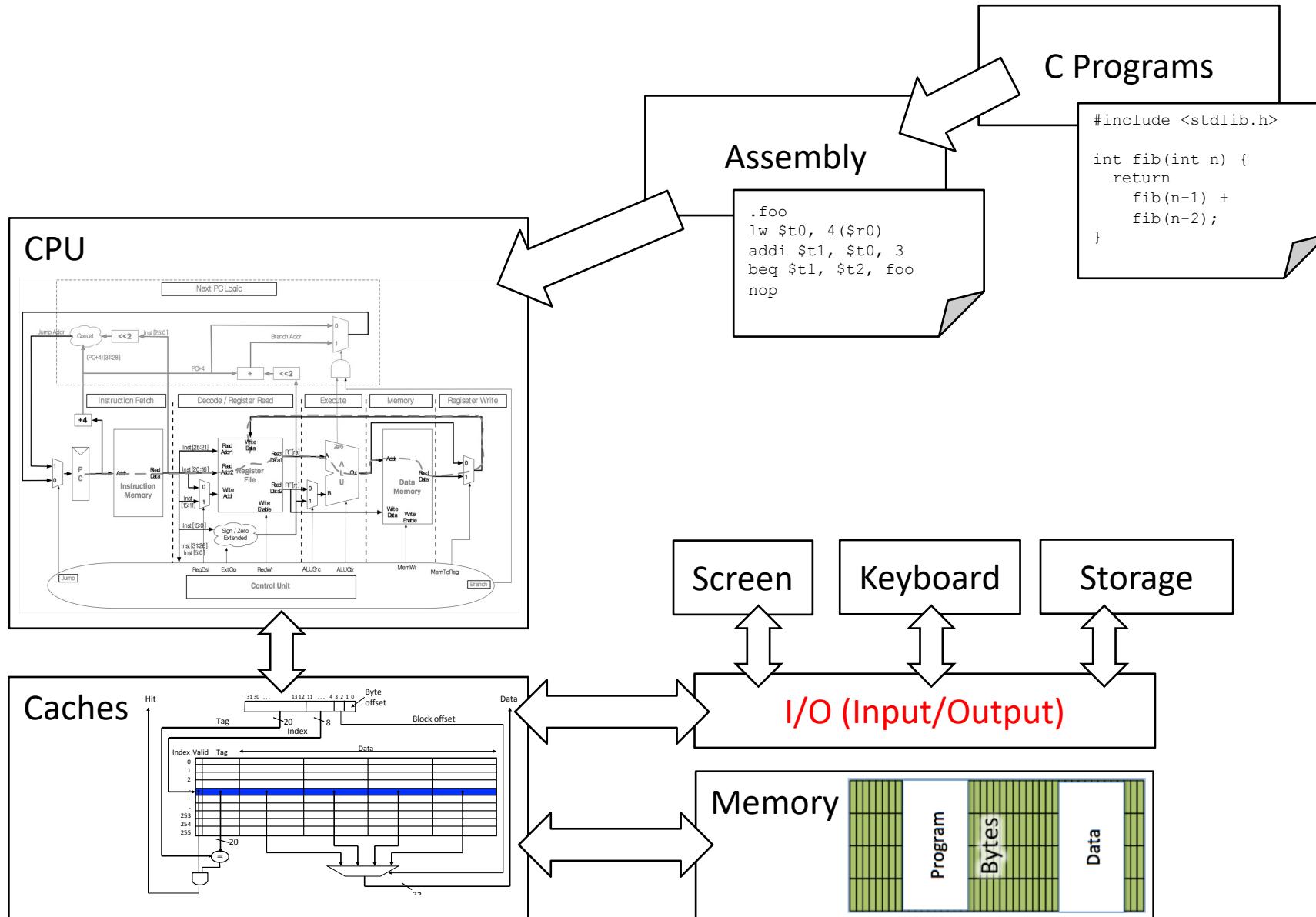
Mémoire Virtuelle



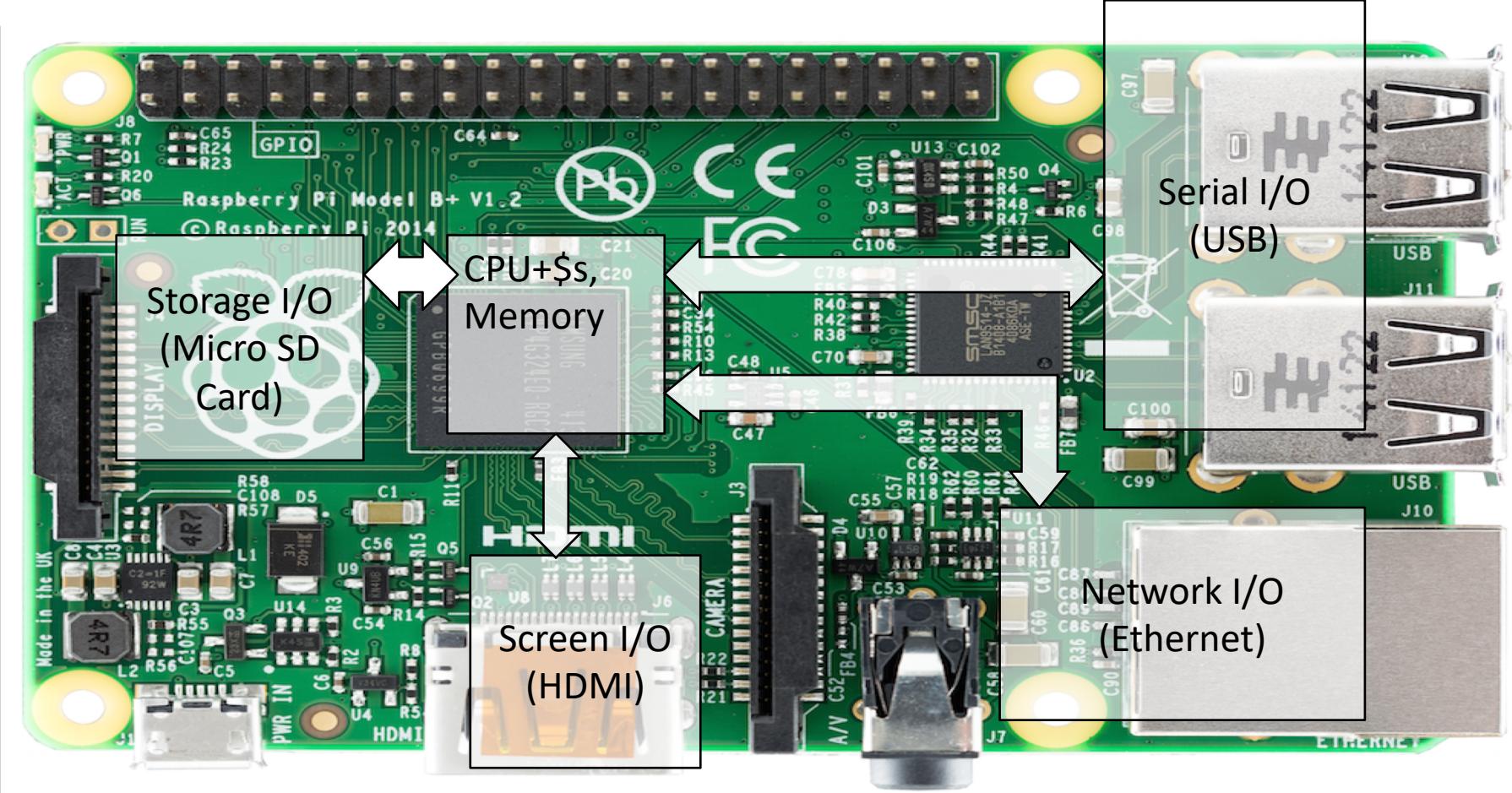
Des differences avec votre PC?



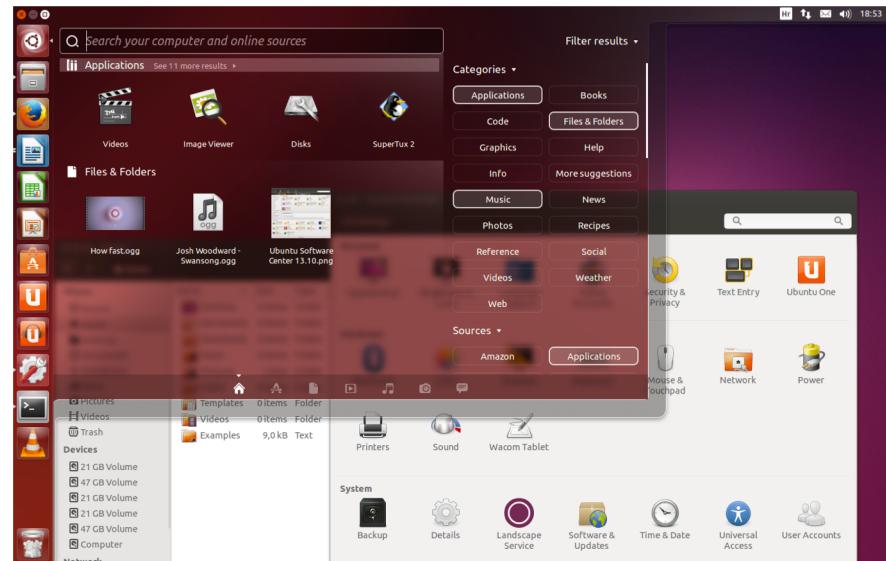
I/O



Raspberry Pi



- Sur un PC plusieurs applications sont exécutées simultanément...



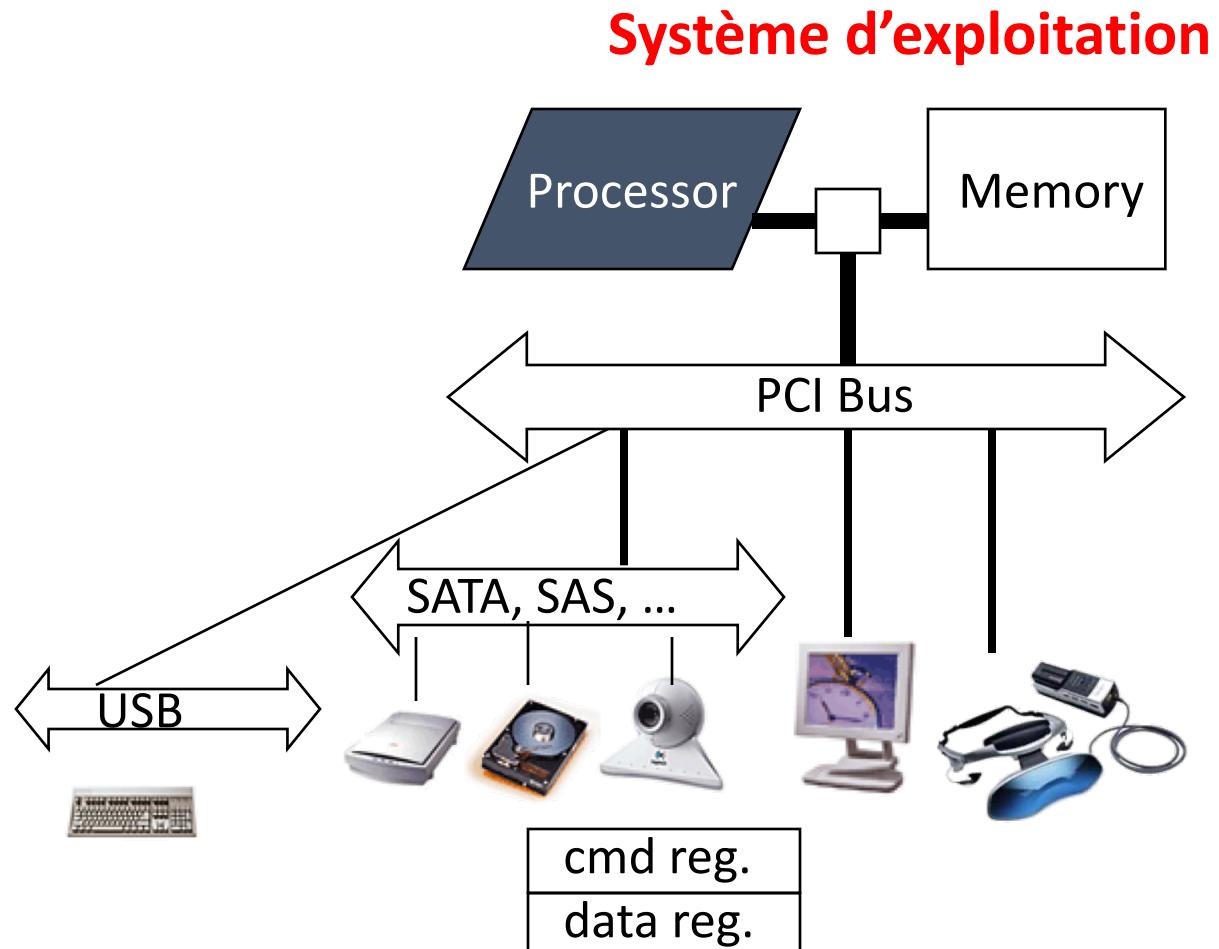
Ceci grâce au système d'exploitation (Operating System (OS))

Que fait l'OS?

- L'OS est la première chose exécutée lorsqu'un PC démarre
- Identifie et contrôle tous les périphériques d'une manière générique
 - S'appuyant sur les "drivers" qui sont dépendant du matériel
- Démarrer les services
 - File system,
 - Network stack (Ethernet, WiFi, Bluetooth, ...),
 - ...
- Charge, exécute et gère les programmes:
 - Plusieurs programmes s'exécutent simultanément(time-sharing)
 - Isole les programmes des uns des autres(isolation)
 - Multiplexe les ressources entre les applications (e.g., périphériques)

Comment intéragir avec les périphériques?

- Comment un programme qui s'exécute interagit avec son environnement?
- On utilise les I/Os.

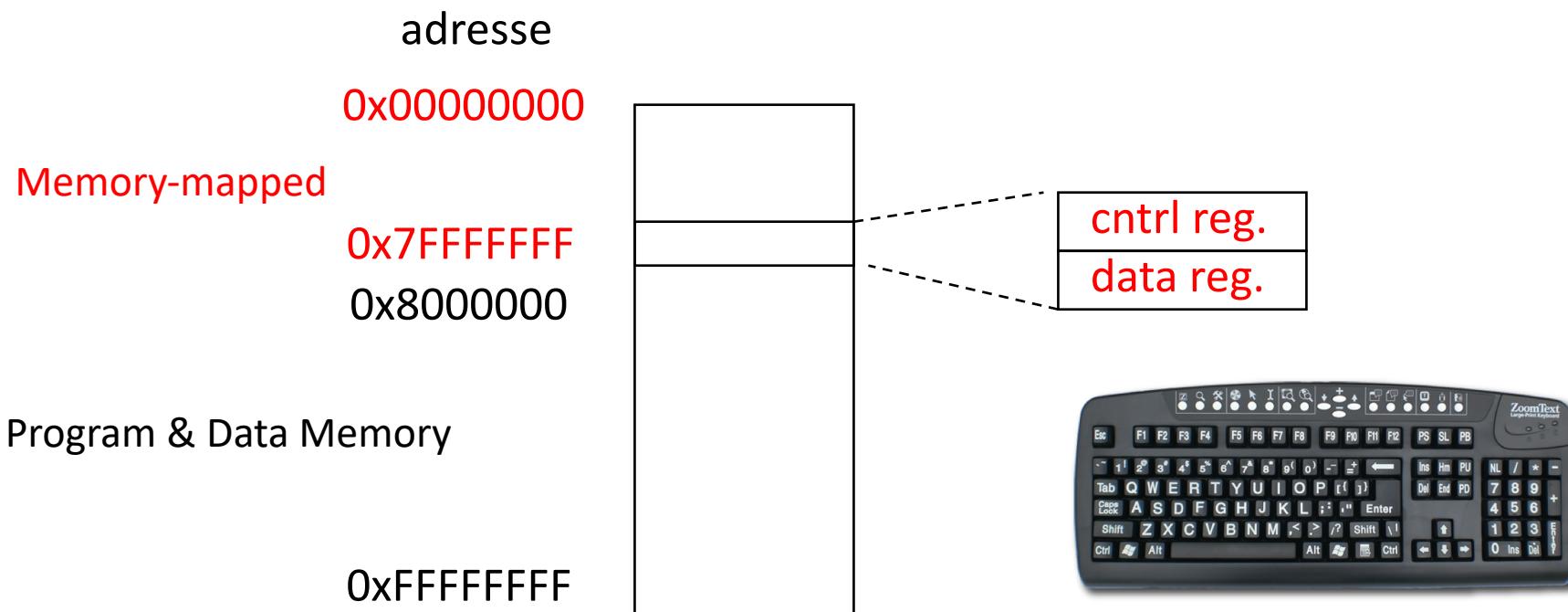


Instruction Set Architecture for I/O

- Que doit faire le processeur pour échanger avec les I/O?
 - Entrée: lire une séquence d'octets
 - Sortie: Ecrire une séquence d'octets
- 2 options
 - a) Des instructions dédiées
 - b) Des I/Os intégrées dans le plan mémoire et donc adressables
 - Une portion de l'espace d'adressage est dédiée aux I/Os
 - on utilise les instructions load/store pour accéder aux registres des périphériques qui ont chacun une adresse

I/O dans l'espace d'adressage

- Certaines adresses ne correspondent pas à des zones de la mémoire
- Elles correspondent à des registres des périphériques.

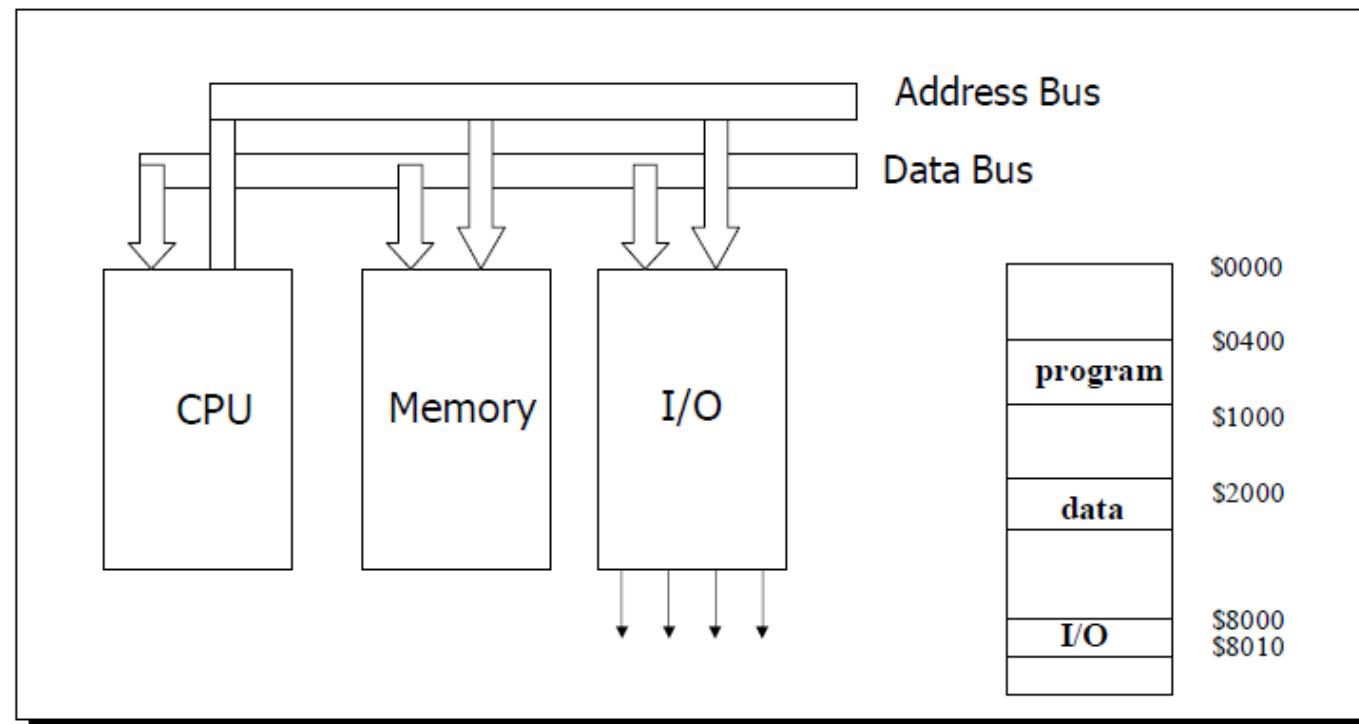


Comment communiquer avec un périphérique?

- Les périphériques peuvent faire partie de l'espace mémoire du processeur
 - Par exemple, l'instruction MOVE.B \$8000,D0 peut permettre de charger un octet provenant d'un périphérique et non de la mémoire.
- Pour accéder à un périphérique il y a plusieurs stratégies:
 - Polling
 - Interruption
 - DMA (direct memory access)

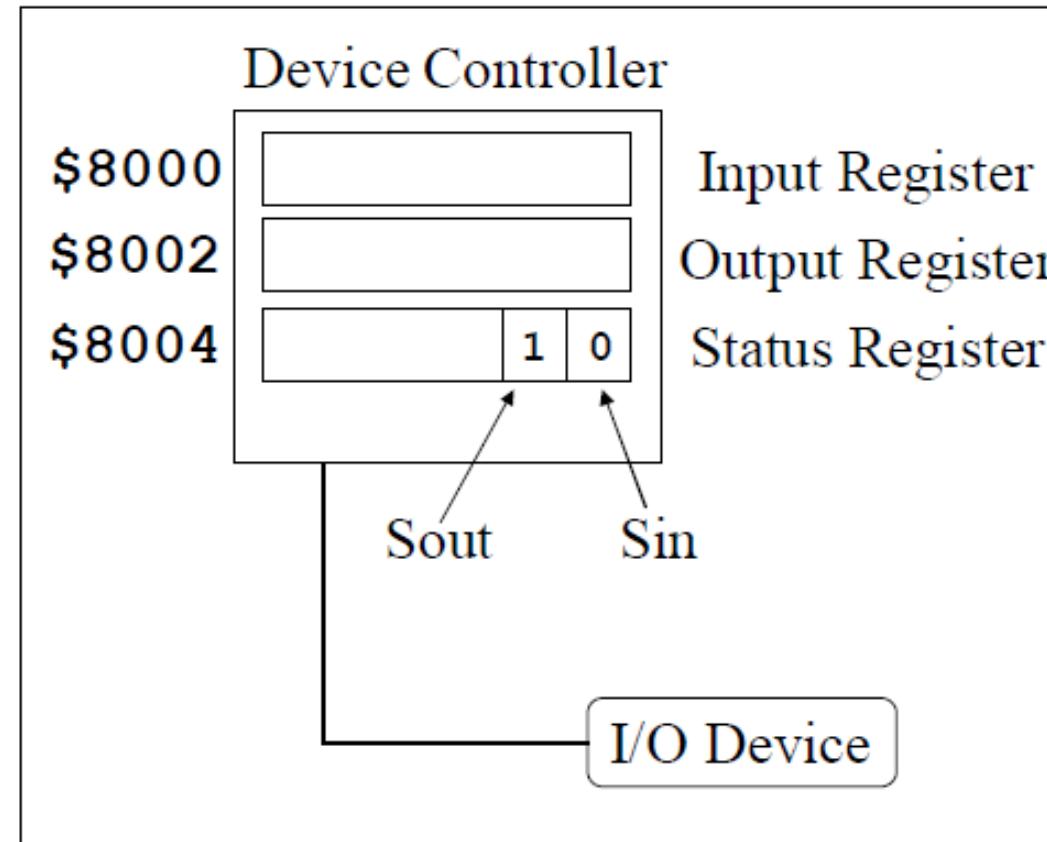
Périphérique dans l'espace mémoire

- Les périphériques sont connectés aux bus d'adresse et de donnée et font donc partie de l'espace mémoire du système.



Exemple de périphérique: périphérique de communication

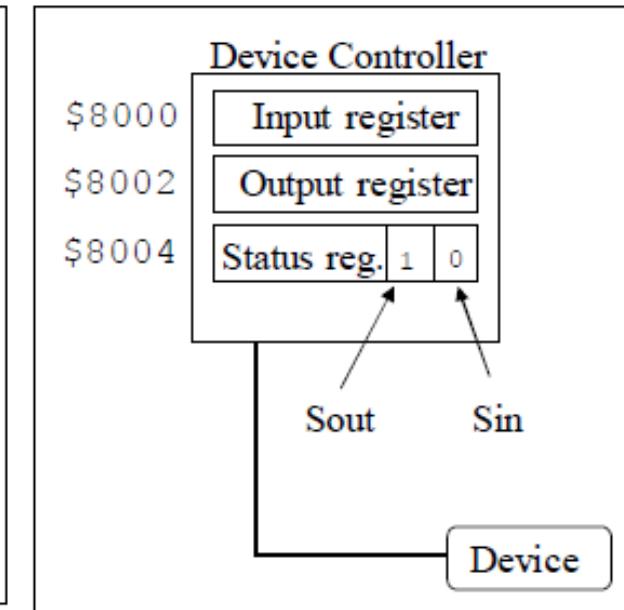
- L'adresse de base est assignée par le concepteur matériel
- Registre de statut:
 - Sin un mot est disponible
 - Sout le périphérique est prêt pour transmettre



Le polling

- Le périphérique contient un registre d'état.
- Le processeur vient continuellement lire ce bit pour connaître l'état du périphérique et effectuer les actions nécessaires.

GETCH	LEA	BUFFER,A0
POLLI	MOVE .B	\$8004 ,D0
	BTST .B	#0 ,D0
	BEQ	POLLI
	MOVE .B	\$8000 , (A0) +
PUTCH	MOVE .B	\$8004 ,D1
	BTST .B	#1 ,D1
	BEQ	PUTCH
	MOVE .B	D0 ,\$8002



Ex: gestion de la communication par polling

Inconvénient du polling

- Le CPU ne peut rien faire d'autre
- Chaque périphérique doit être explicitement interrogé
 - Il faut mettre en place un ordonnancement
- Il n'est pas possible d'attirer l'attention du cpu pour un périphérique tant que ce n'est pas son « tour ».
 - Pas de réponse rapide à un événement externe

Interruptions

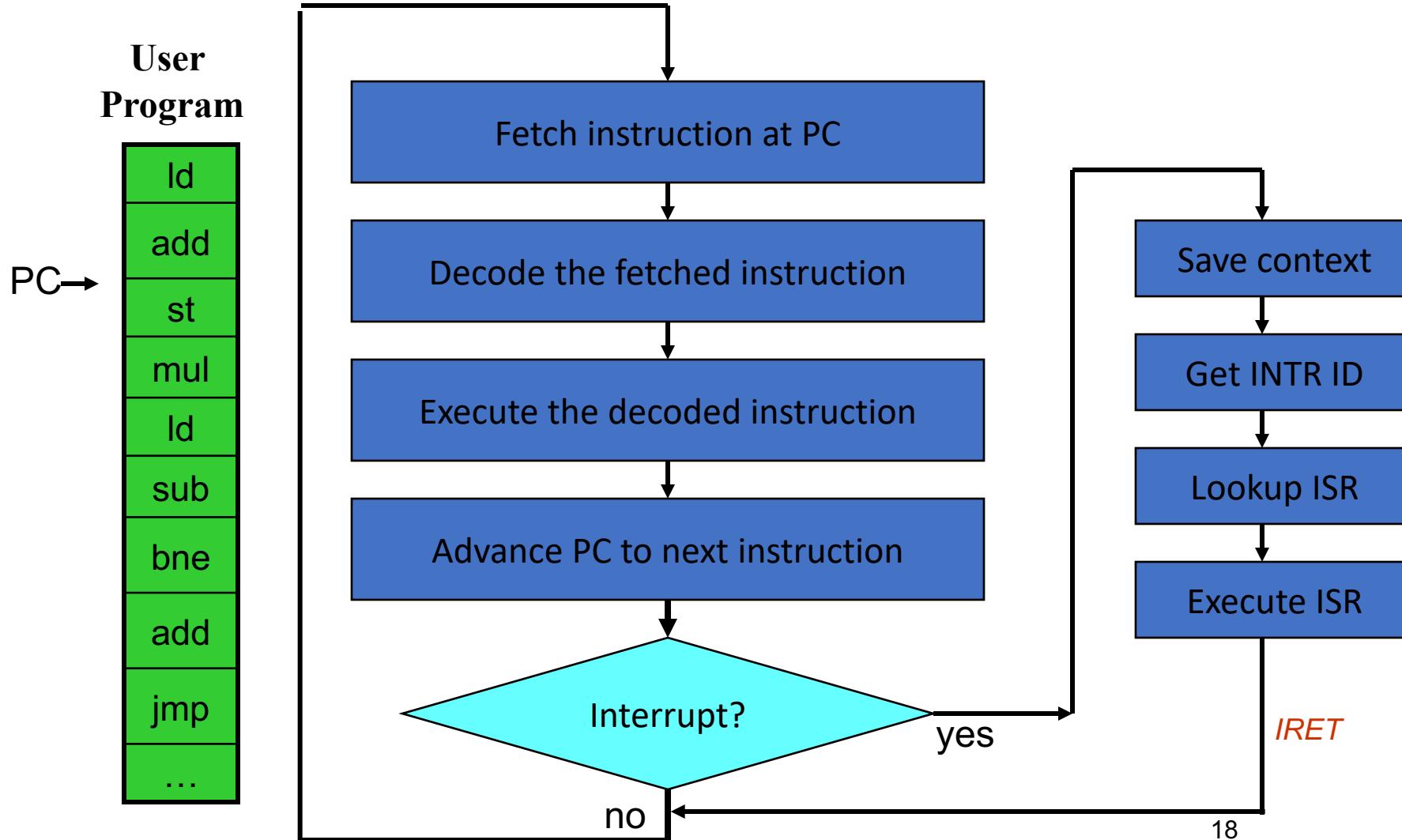
- L'intérêt d'un processeur dépend de sa capacité à interagir avec les périphériques (clavier, écran, réseau..)
- Les périphériques nécessitent une réponse rapide du processeur lorsque certains événements se produisent même si le processeur est occupé par l'exécution d'une tâche.
- On a donc besoin d'un mécanisme pour « attirer l'attention » du processeur.

=> Les interruptions

Interruptions

- Change le flot de contrôle du programme
- Similaire à un changement de contexte
 - Le processeur sauvegarde quelques informations dans la pile
 - Donne la main au noyau, celui-ci décode quel « handler » (ou ISR) d'interruption doit être exécuté
 - On reprend l'exécution avec une instruction dédiée « rte »
- Il y a différents types d'interruptions (matérielles, logicielles)

Cycle d'exécution d'un programme



Les modes différents modes

Mode Superviseur

- Si cela se passe mal pour une application, cela peut engendrer un crash de la machine...
- Le système d'exploitation permet de limiter les ressources d'une application (accès à la mémoire aux périphériques...)
- Pour protéger l'OS des applications, les processeurs ont un mode superviseur:
 - Un processus peut accéder seulement à un sous ensemble des instructions et à une partie de la mémoire physique lorsqu'il n'est pas en mode superviseur (c'est le mode utilisateur, user mode)
 - Pour passer en mode superviseur, il y a des instructions dédiées

Appels systèmes: Syscalls

- Comment faire si on veut appeler une routine de l'OS?
 - Pour lire un fichier
 - Lancer un nouveau processus
 - Demander plus de mémoire (malloc)
 - Envoyer des données...
- Besoin d'exécuter un appel système **syscall**:
 - Les arguments sont placés dans des
 - On lève une interruption logicielle (instructions assembleur dédiées)
- L'OS exécute les opérations et rend la main à l'application en mode user
- De cette façon, l'OS peut contrôler tous les accès aux ressources critiques.

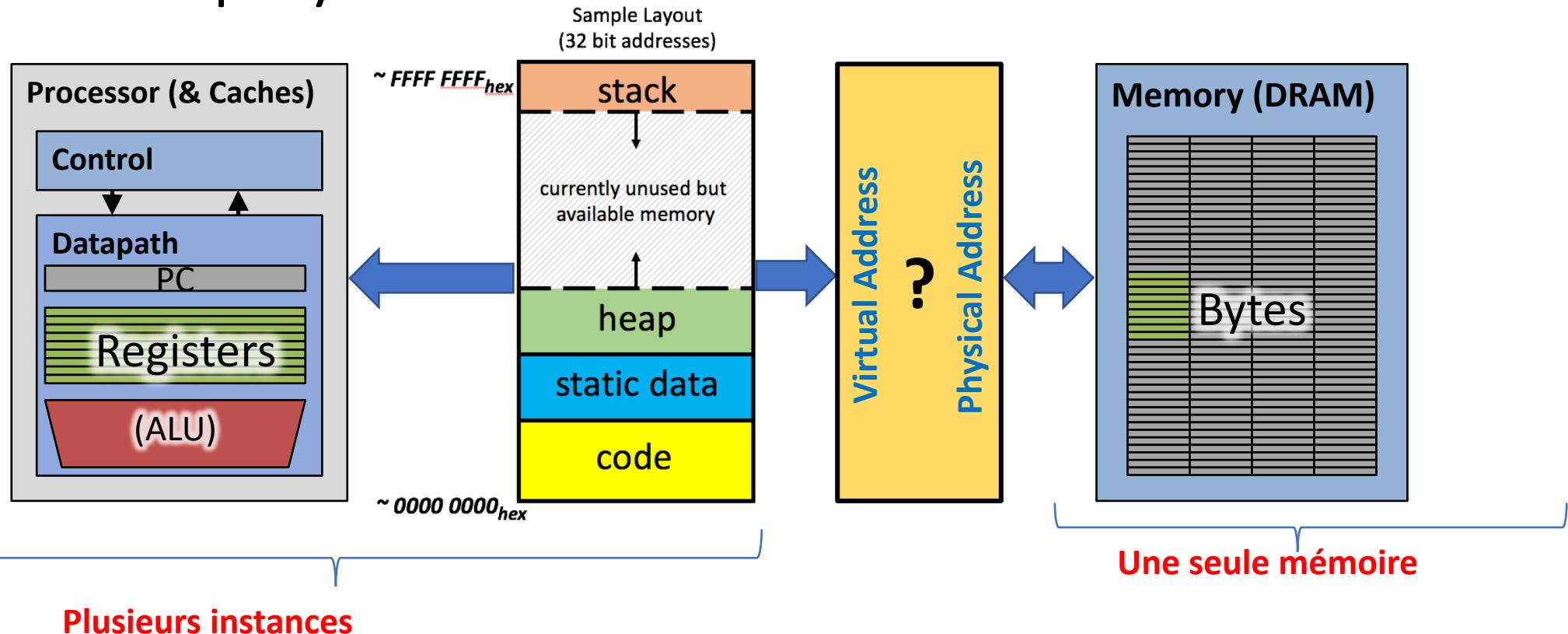
Multiprogramming

- L'OS exécute plusieurs applications simultanément.
- Pas tout à fait (à moins d'avoir un cœur par processus)
- Il commute entre les processus très rapidement (commutation de contexte “context switch”)
- Lorsqu'il saute dans un nouveau processus, il fixe une interruption timer
 - Lorsque le timer expire, on stocke le PC, les valeurs des registres...(état du processus)
 - Choisit un nouveau processus à exécuter et charge son état dans le processeur
- L'ordonnancement (ou scheduling) permet de décider quel processus doit être exécuté.

Protection, Translation, Paging

- Le mode superviseur seul n'est pas suffisant pour isoler les applications entre elles ou du système d'exploitation.
 - Une application pourrait modifier une zone mémoire d'une autre application
 - Typiquement, l'adresse de départ d'un programme est fixée par convention (e.g. 0x8FFFFFFF), comment plusieurs programmes peuvent avoir la même adresse de départ?
 - Aussi, on pourrait avoir besoin d'adresser plus de mémoire que ce que l'on dispose physiquement
- Une solution: **La mémoire Virtuelle**
 - Donne à chaque processus l'illusion d'un espace d'adressage complet qui lui est totalement dédié.

Adresses physique vs. Adresses virtuelles



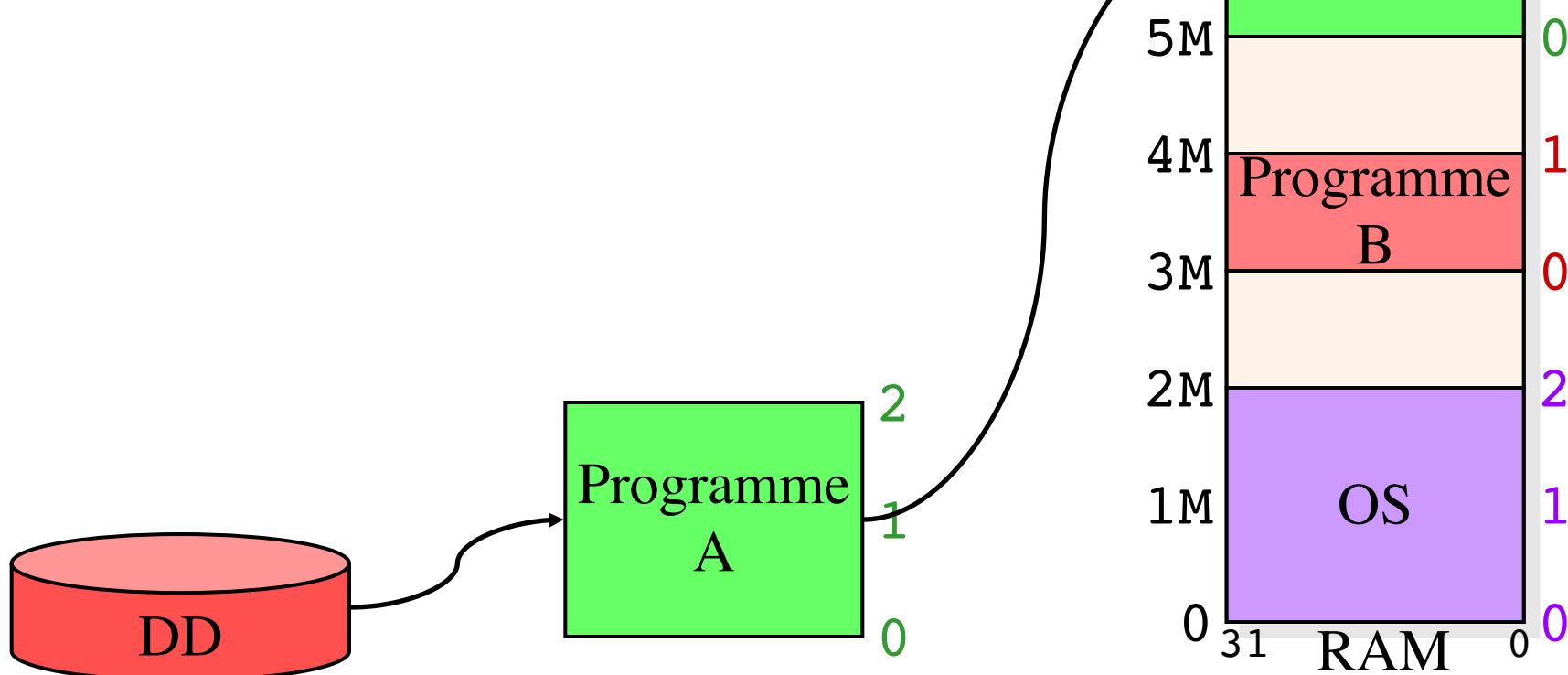
- Les processus utilisent des adresses virtuelles, e.g., 0 ... 0xffff,ffff
 - Plusieurs processus peuvent utiliser les mêmes adresses (possibilité de conflit)
- La mémoire utilise des adresses physiques (0 ... 0xffff,ffff)
- **La MMU (Memory Management Unit) traduit chaque adresse virtuelle vers une adresse physique**

Espace d'adressage

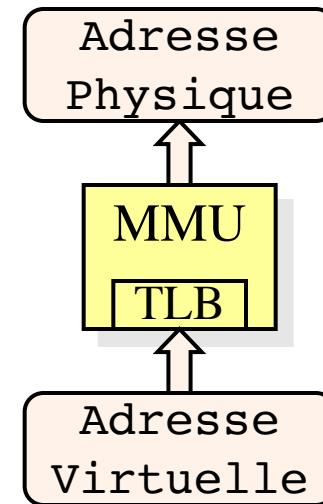
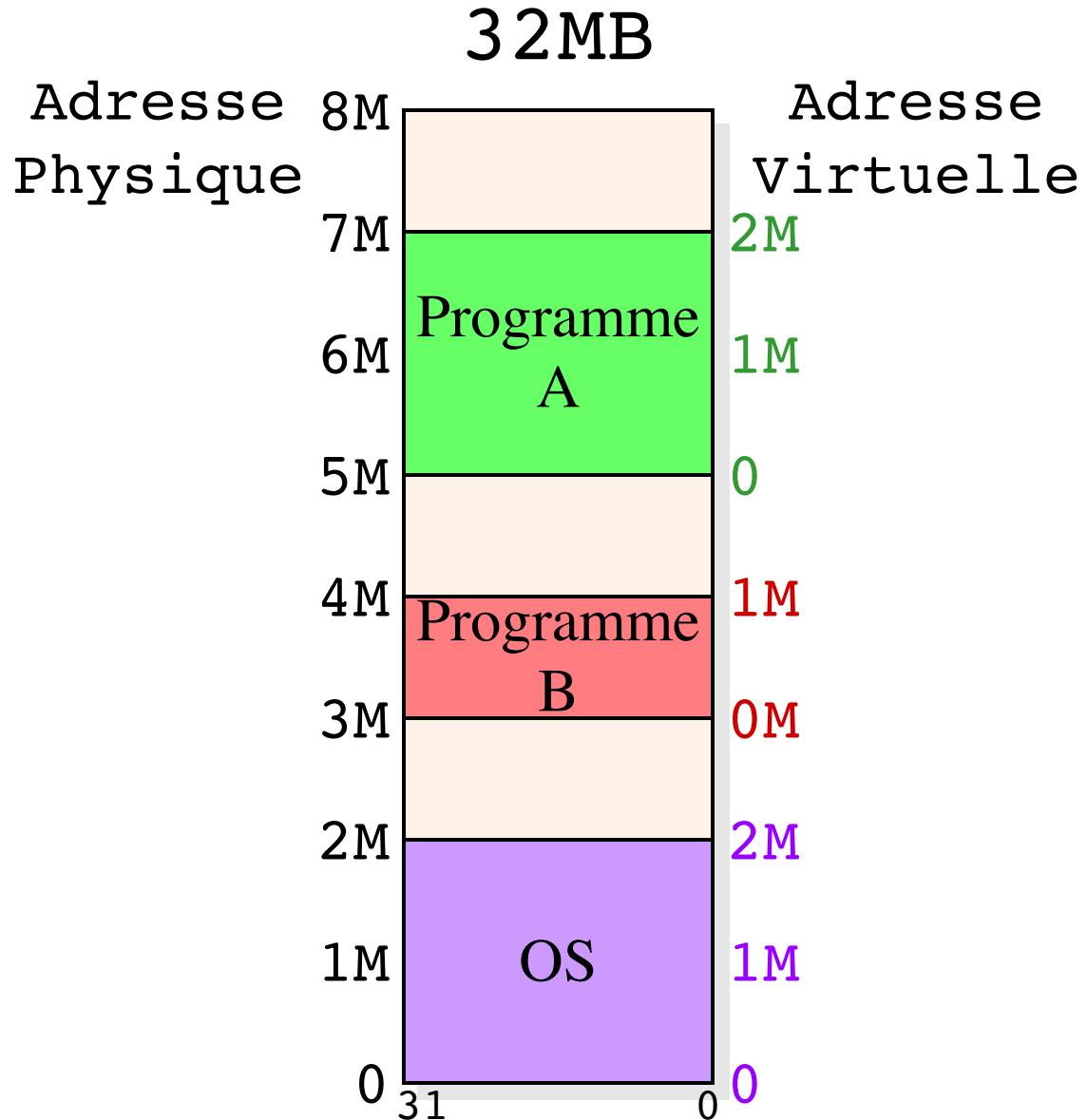
- Espace d'adressage= ensemble des adresses pour tous les emplacements disponibles en mémoire.
- 2 types d'espace d'adressage:
 - **Espace d'adressage virtuel**
 - Ensemble des adresses qu'une application voit
 - **Espace d'adressage physique**
 - Ensemble des adresses qui correspondent aux emplacements physiques en mémoire
 - Non visible des applications
 - La MMU (Memory Management Unit) fait le lien entre les deux espaces.

Espace d'adressage

Dans le cas **simple** d'un programme qui peut être stocké entièrement en mémoire, l'OS se charge de le transférer du disque dur à la mémoire RAM, en lui réservant un **espace suffisant**. L'emplacement du programme dans la RAM peut **varier** d'un lancement à l'autre. Toutes les références internes au programme (sauts) se font par **déplacement**.



Mémoire virtuelle



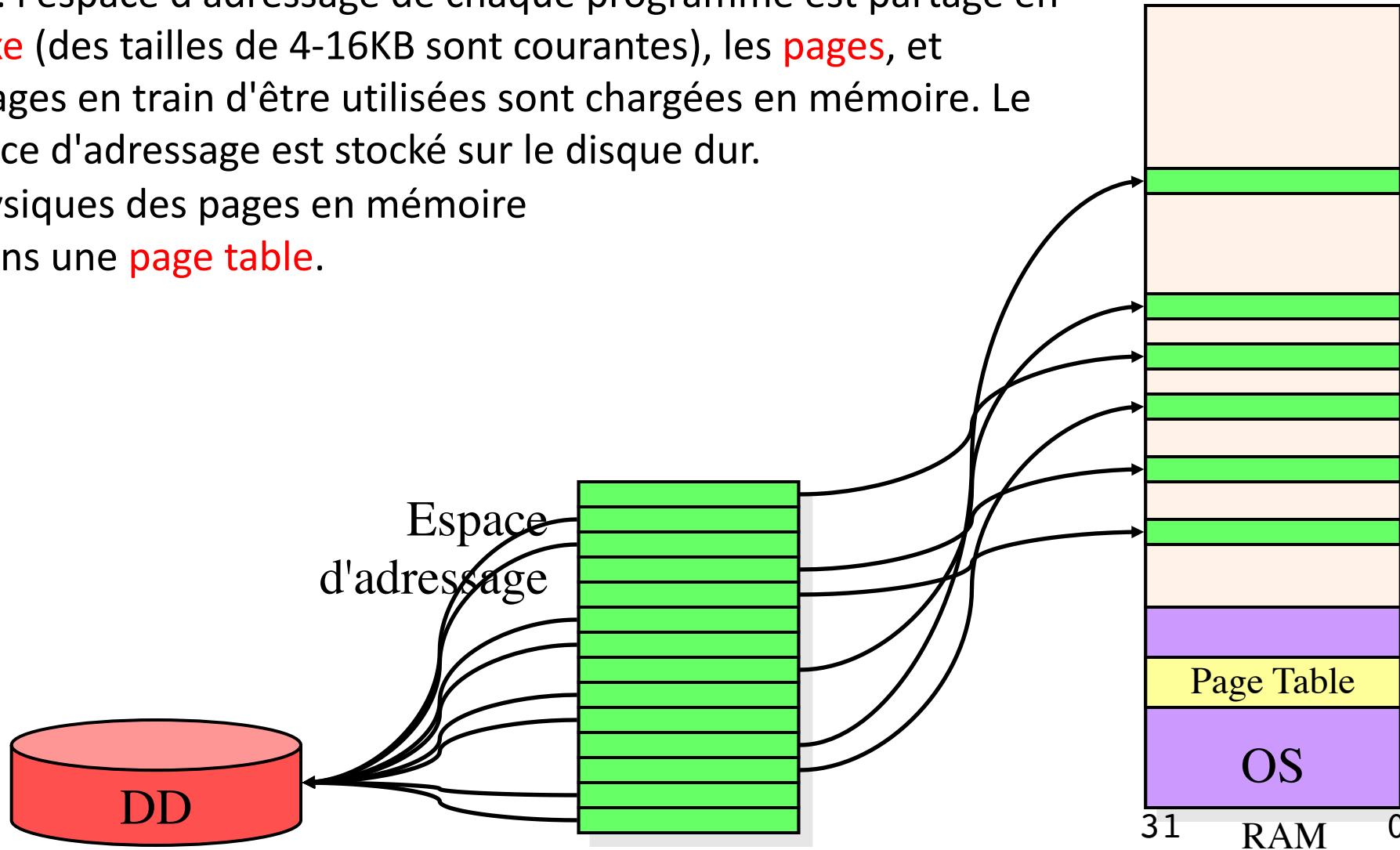
Mémoire virtuelle

- Lors du lancement d'un programme, l'OS réserve un **espace d'adressage virtuel** pour le programme. Cet espace a une **taille variable**, déterminée par le programme lui-même lors de sa compilation (mais limitée par l'OS).
- Toute référence à la mémoire est faite par rapport à cet espace d'adressage virtuel, et non pas par rapport à la mémoire physique. La mémoire physique héberge tous les espaces virtuels de tous les programmes en train de s'exécuter sur le processeur.
- Une unité dans le processeur (le **MMU** ou *memory management unit*) se charge de traduire les adresses virtuelles en adresses physiques.

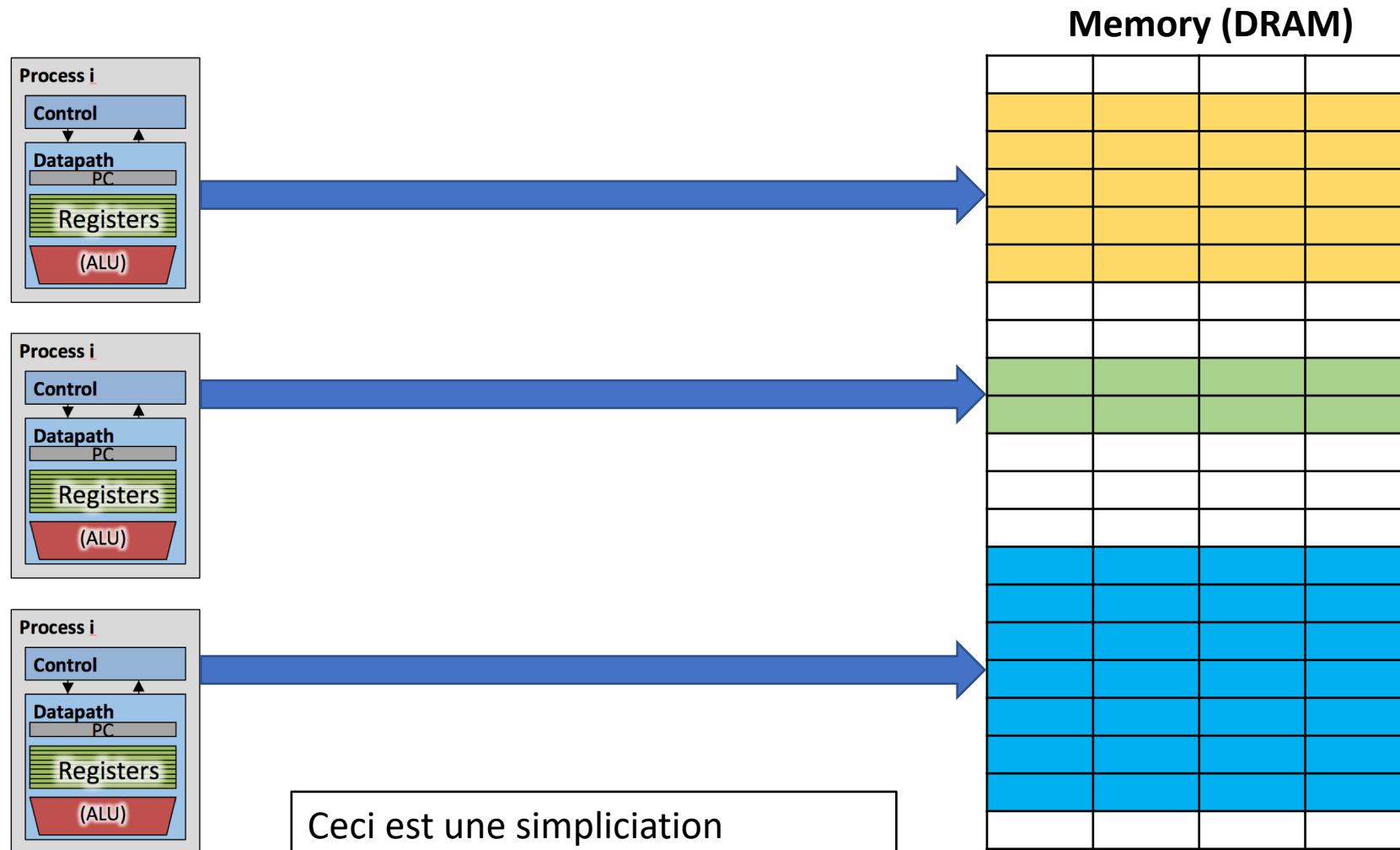
Pagination

Le principe de la pagination est simple (sa mise en œuvre, par contre, est assez complexe): l'espace d'adressage de chaque programme est partagé en blocs de **taille fixe** (des tailles de 4-16KB sont courantes), les **pages**, et seulement les pages en train d'être utilisées sont chargées en mémoire. Le restant de l'espace d'adressage est stocké sur le disque dur.

Les adresses physiques des pages en mémoire sont stockées dans une **page table**.



Vision conceptuelle de la gestion de la memoire



Rôle de la MMU

- 1) Faire correspondre des adresses virtuelles aux adresses physiques
- 2) Protection:
 - Isoler les zones mémoires entre les processus
 - Chaque processus a sa propre zone mémoire
 - Des erreurs dans un programme ne viennent ainsi pas corrompre la mémoire d'un autre programme.
- 3) Echanger (“swap”) des données entre la mémoire de travail et le disque
 - Donne l'illusion d'une plus grande mémoire en stockant certaines parties sur le disque
 - LE disque est généralement plus grand et plus lent que la mémoire principale

La MMU

- La pagination (paged memory) est la principale technique utilisée aujourd’hui:
 - La mémoire physique est divisée en page
 - La taille typique des pages est de: 4 KiB+

Adresse virtuelle (32 bits)

Numéro de page

offset

Pagination - Adressage

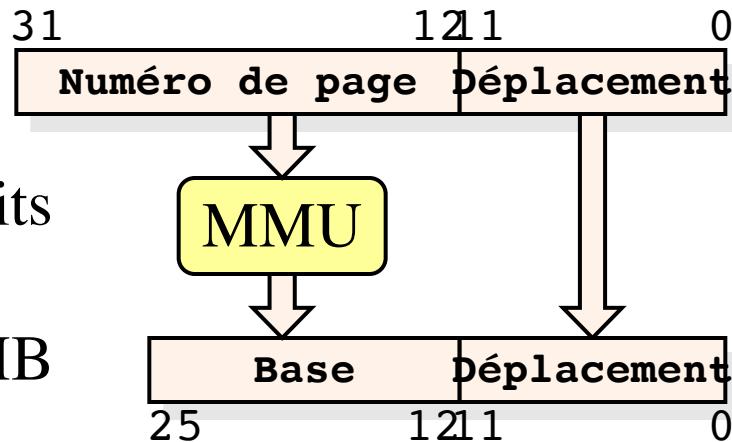
Une adresse virtuelle est donc composée de deux parties: un **numéro de page** et un **déplacement** dans la page.

Exemple:

Largeur des adresses = 32bits

Taille des pages = 4KB

Taille de la mémoire = 64MB



Cette transformation, réalisée par le MMU, permet d'une part de travailler avec des espaces d'adressage plus grands que la mémoire physique, et d'autre part de référencer des données sur une même page avec peu de bits (ce qui permet l'utilisation de certains modes d'adressage).

Pagination - *Page faults*

- Une adresse virtuelle peut donc référencer soit une page en mémoire (*page hit*) soit une page sur le disque dur (*page miss*).
- Si la page cherchée se trouve sur le disque dur, le MMU engendre une interruption (*page fault*) et une procédure se charge de transférer les données en mémoire (une opération qui demande plusieurs millions de cycles d'horloge).
- La taille des pages est un paramètre fondamentale dans la minimisation des *page faults*: une taille trop petite augmente le nombre de *page miss*, et une taille trop grande augmente la pénalité associé au transfert des données. Dans les systèmes actuels, la taille des pages varie entre 4 et 16KB.

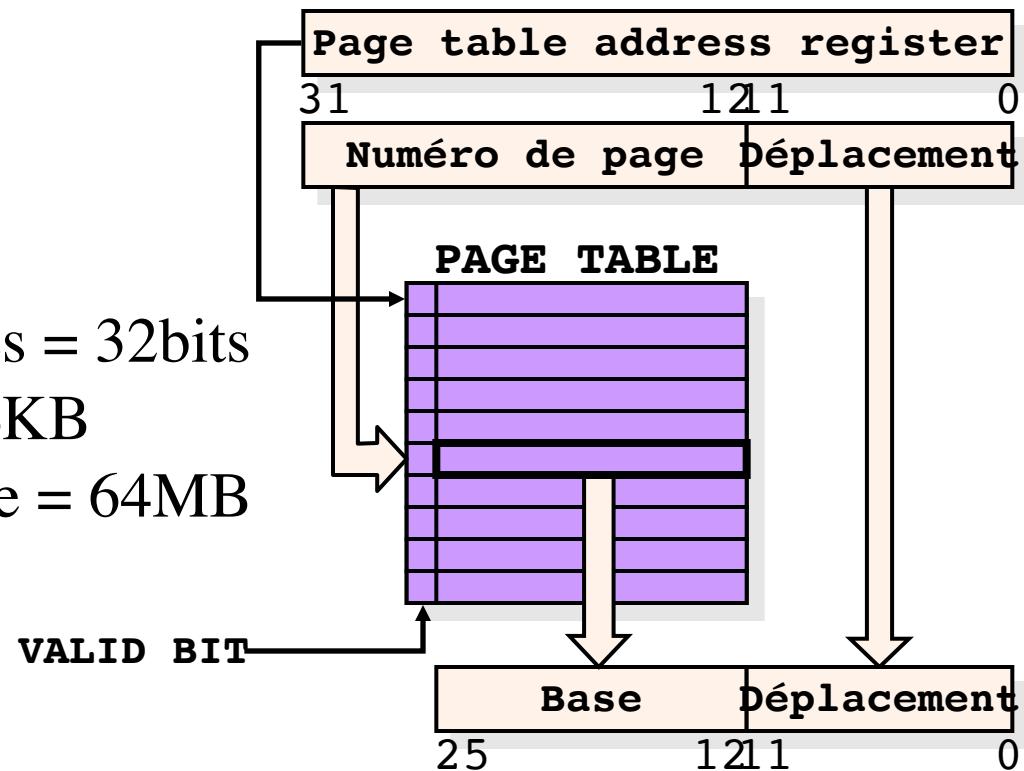
Pagination - Remplacement

- Lors qu'une nouvelle page est transférée du disque dur à la mémoire, il est parfois nécessaire d'en transférer une autre de la mémoire au disque dur (*swap*).
- Étant donné le "prix" très élevé d'un *page fault*, l'algorithme utilisé pour décider quelle page va être "swappée" est fondamental pour la performance d'un système. L'algorithme de choix pour cette opération est le **LRU** (*least recently used*).
- Si la mémoire est trop petite par rapport aux besoins d'un programme, ou si l'algorithme de swap n'est pas efficace ou si les données sont "mal" stockées, le phénomène appelé *thrashing* peut se produire: la machine passe tout son temps à transférer des pages et n'arrive pas à calculer.

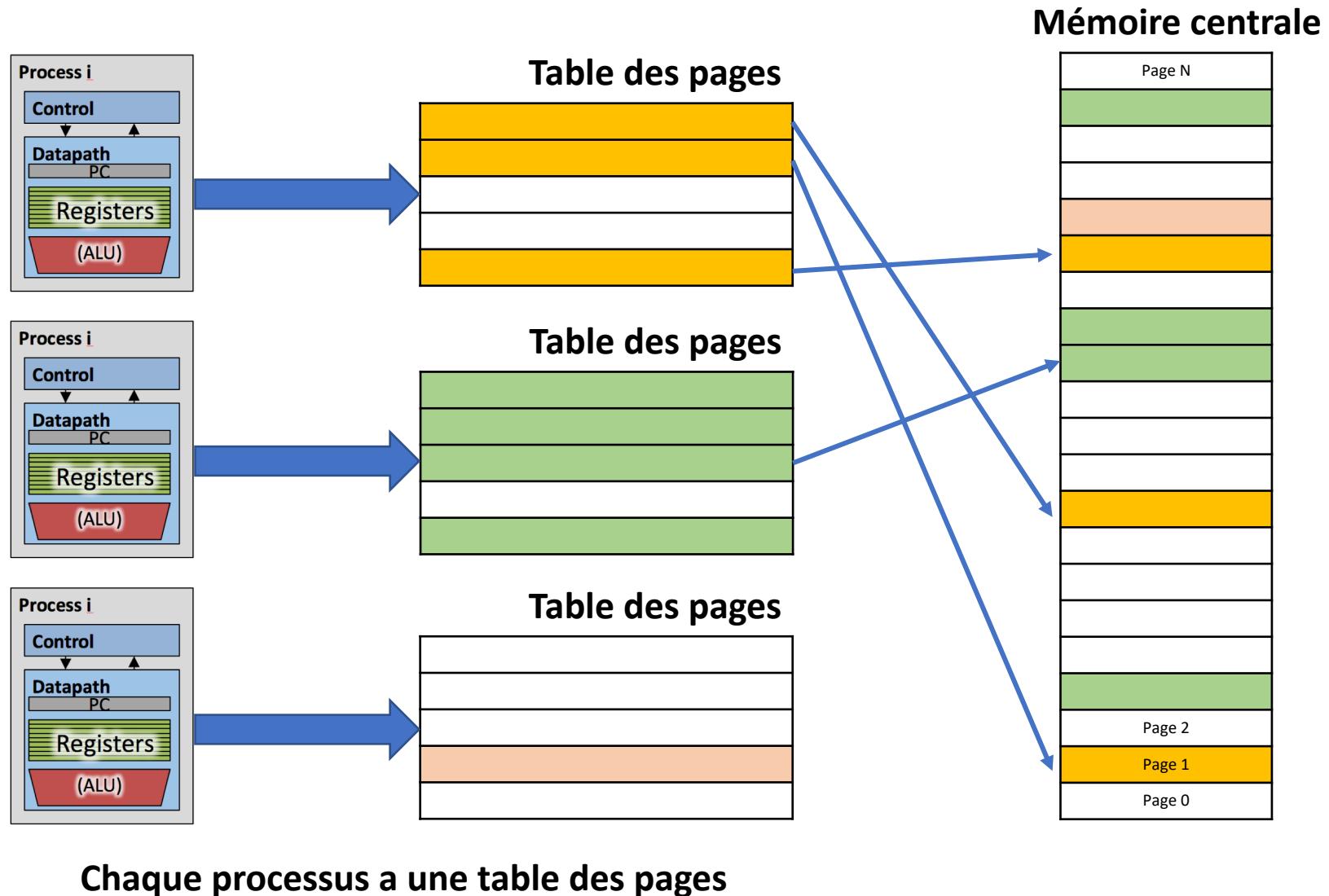
Pagination - Implémentation

- Le MMU doit trouver l'adresse de la page physique correspondant à une page virtuelle. Pour effectuer cette opération, il consulte un **tableau de pages (*page table*)**, stocké en **mémoire** et géré par l'OS.

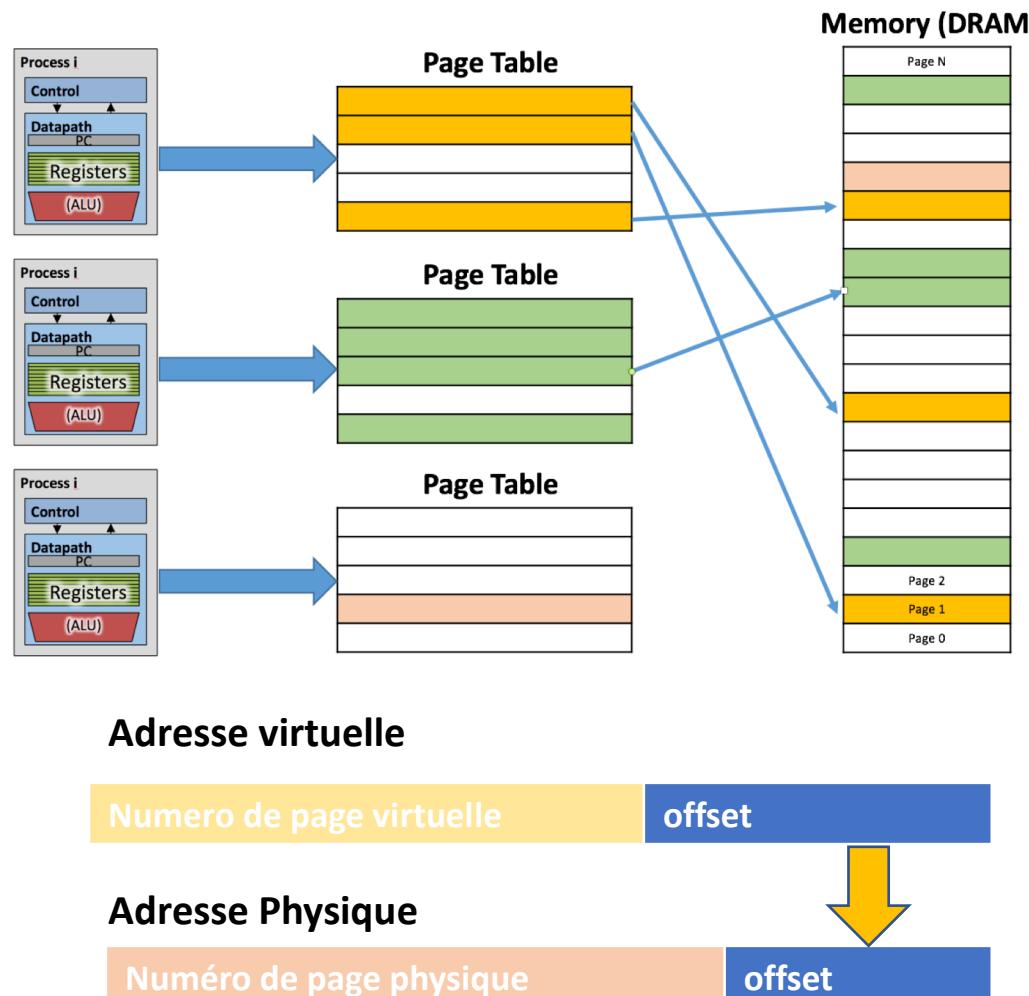
Exemple:
Largeur des adresses = 32bits
Taille des pages = 4KB
Taille de la mémoire = 64MB



Pagination

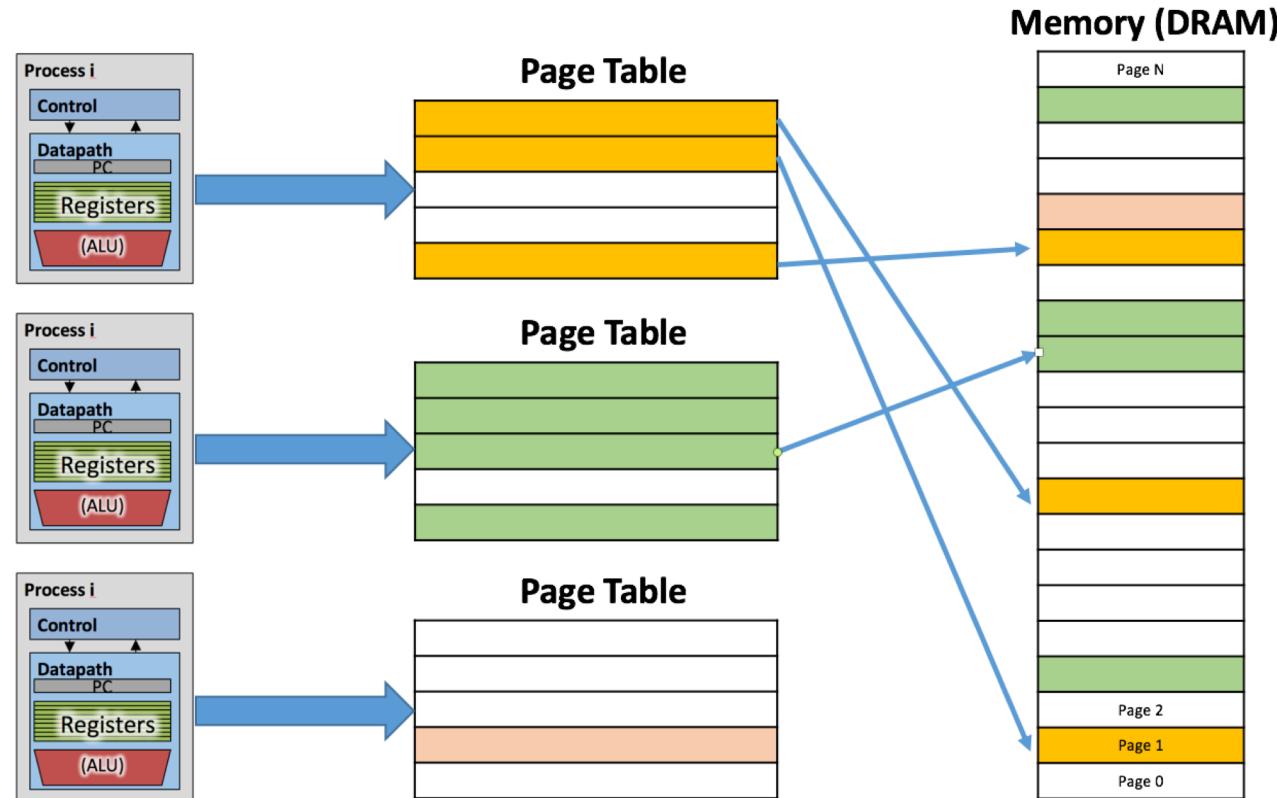


Translation des pages virtuelles vers les pages physiques



- Le système d'exploitation a connaissance du processus en cours d'exécution
 - Il sélectionne donc la table des pages correspondant
- La MMU extrait le numéro de page virtuel depuis l'adresse virtuelle
- Cherche la page physique correspondant dans la table
- Génère l'adresse physique correspondant en concaténant:
 - Le numéro de page physique
 - Et l'offset

Protection



- Assigner des pages différentes aux processus permet de les protéger entre eux.
 - Isolation
 - Les tables des pages sont gérés par l'OS (en mode superviseur)
- Il est également possible de partager des pages
 - L'OS peut assigner des pages physiques identiques à plusieurs processus.

Où se trouvent les tables des pages?

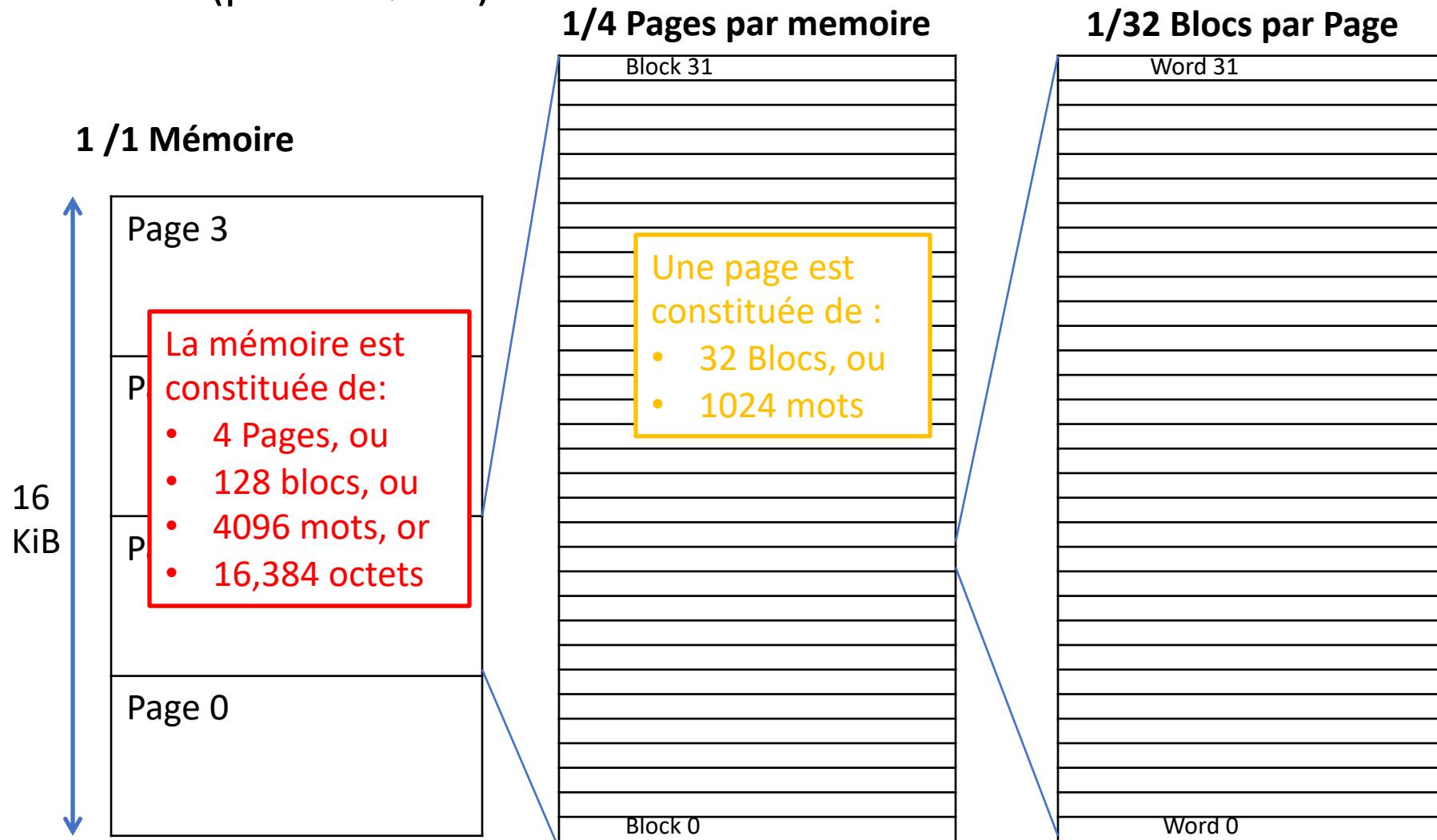
- Si on a des adresses virtuelles sur 32 bits et des pages de 4KiB
 - Taille de la table des pages:
 - 4×2^{20} Bytes = 4-MiB
 - 0.1% d'une mémoire de 4-GiB
 - Trop grand pour une mémoire cache
 - Les tables sont stockées en mémoire(DRAM)
 - Comment minimiser le temps d'accès à la table des pages?
 - On utilise un cache pour sauvegarder les correspondances pages virtuelles-pages physiques les plus fréquentes...

Blocs vs. Pages

- En cache on utilise des *blocs*
 - Généralement de taille ~64B dans les processeurs récents
- Pour la mémoire virtuelle, on utilise des *pages*
 - Généralement ~4 KB dans les processeurs récents
- Il ne faut pas mélanger les structures suivantes:
 - Octets -Bytes,
 - Mots - Words,
 - Blocs,
 - Pages
 - Il s'agit simplement de façon différentes de visualiser des parties de la mémoire!

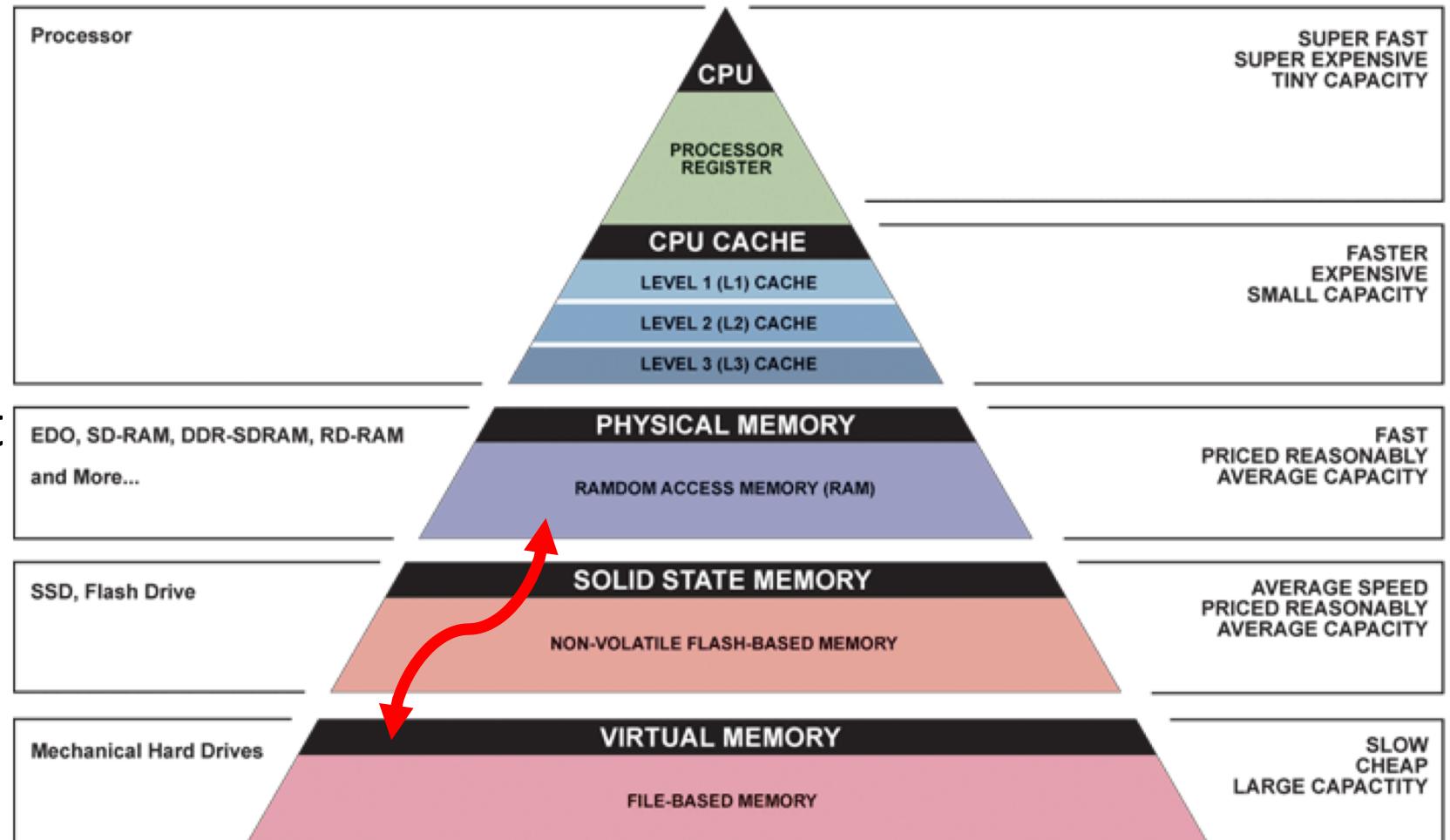
Octets, Mots, Blocs, Pages

E.g.: 16 KiB DRAM, 4 KiB Pages (Mem. Virtuelle), 128 B blocs (Mem. Cache),
4 B words (pour **lw / sw**)



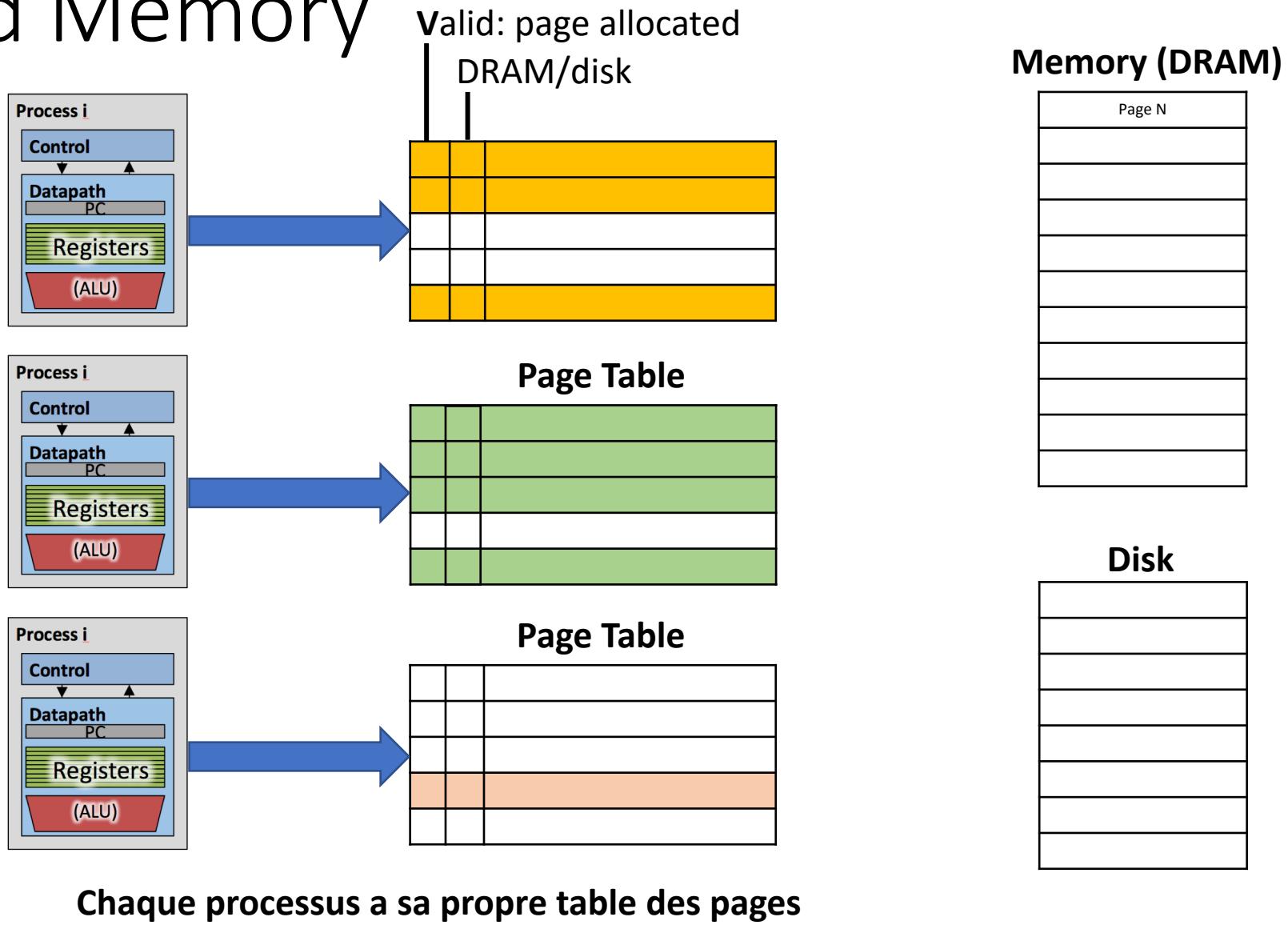
Hiérarchie Mémoire

- Disque
 - lent
 - grande capacité
 - Comment utiliser cette capacité (lorsque la RAM est pleine)



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Paged Memory

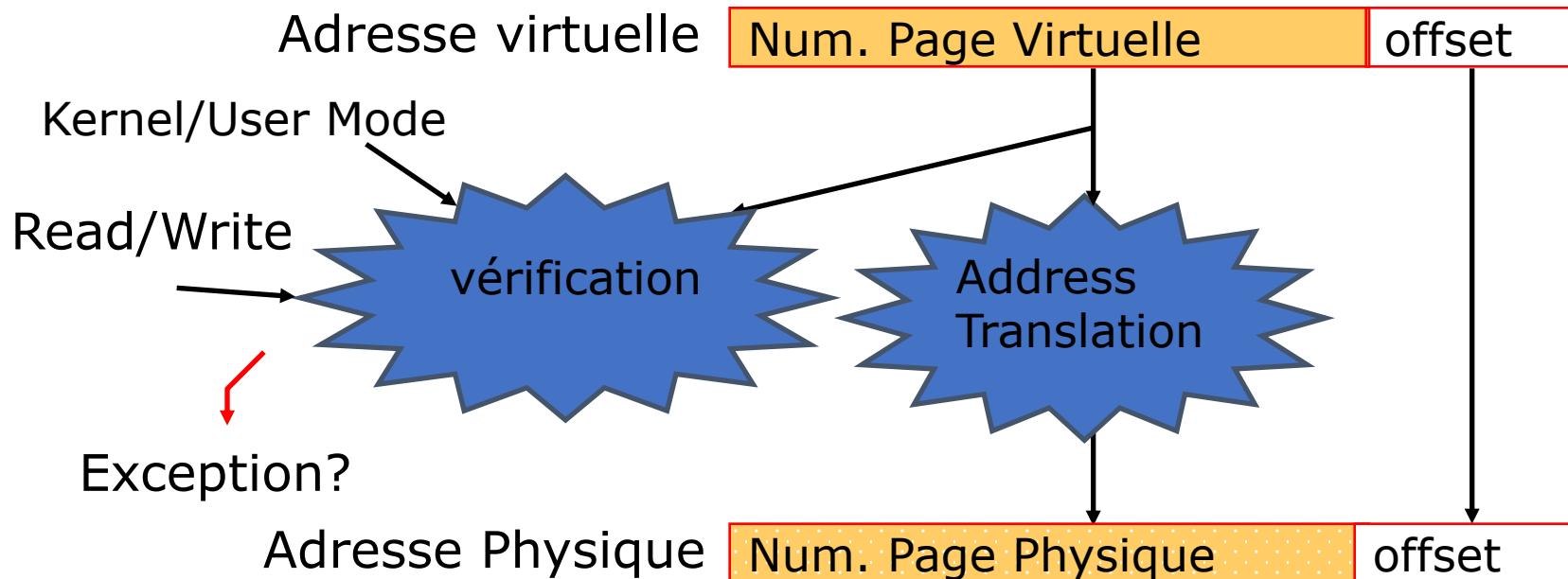


Accès Mémoire

- On vérifie la table des pages:
 - La page est elle valide?
 - Si oui → Est elle en DRAM?
 - Si oui, accès à la donnée en DRAM
 - Si non, mais présente dans le disque: on alloue une nouvelle page dans la DRAM
 - Si la mémoire est pleine on expulse une page de la mémoire
 - La page expulsée est stockée dans le disque
 - On charge la page recherchée du disque vers la mémoire
 - Page non valide
 - On alloue une nouvelle page dans la DRAM
 - Si la mémoire est pleine on expulse une page de la mémoire

**Page fault
Intervention de l'OS**

Translation d'adresse et protection

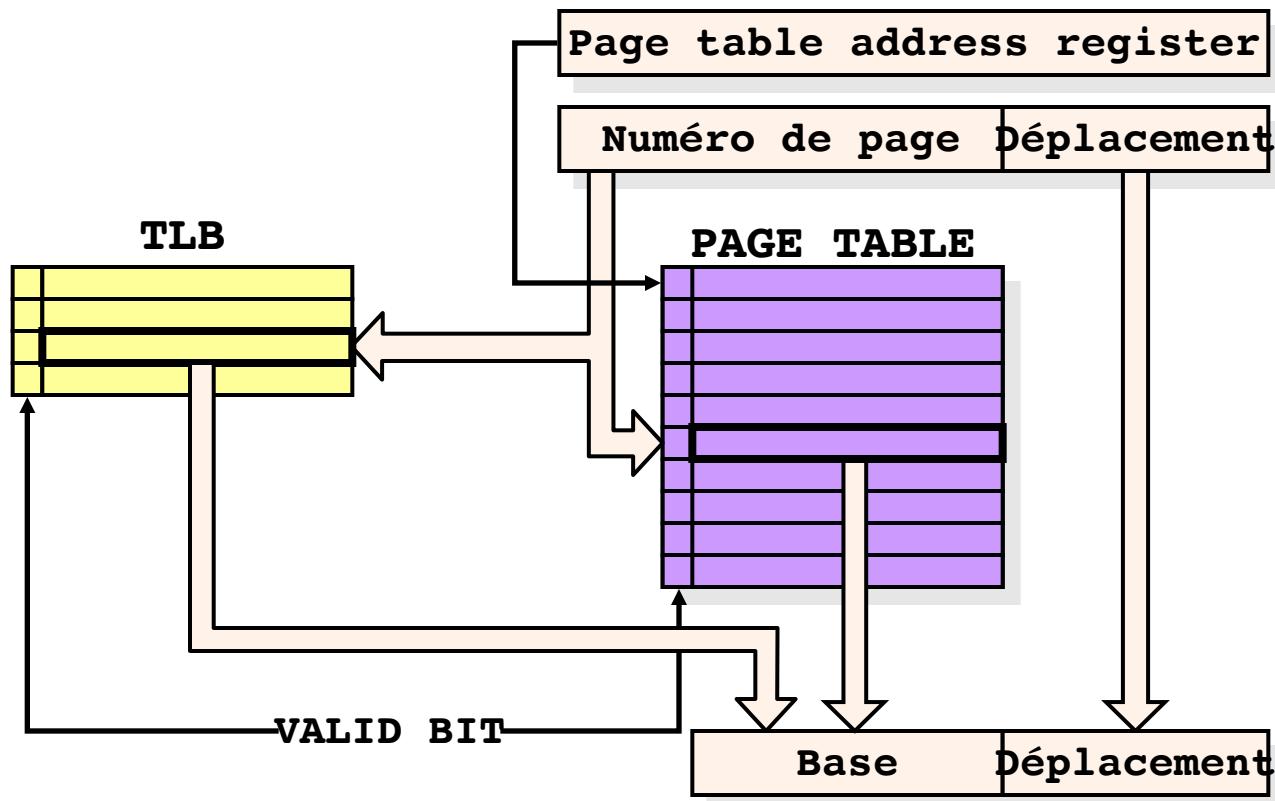


- Chaque accès à une donnée ou une instruction nécessite une translation et une vérification des droits.

Il est nécessaire que ce soit rapide et optimisé

Pagination - TLB

- Pendant l'exécution d'un programme, la même page est souvent référencée plusieurs fois de suite. Pour accélérer la recherche de l'adresse physique, on utilise un "raccourci" matériel: le **TLB** ou **translation lookaside buffer** (une petite table dans le processeur).



Translation Lookaside Buffers (TLB)

La translation d'adresse est coûteuse!

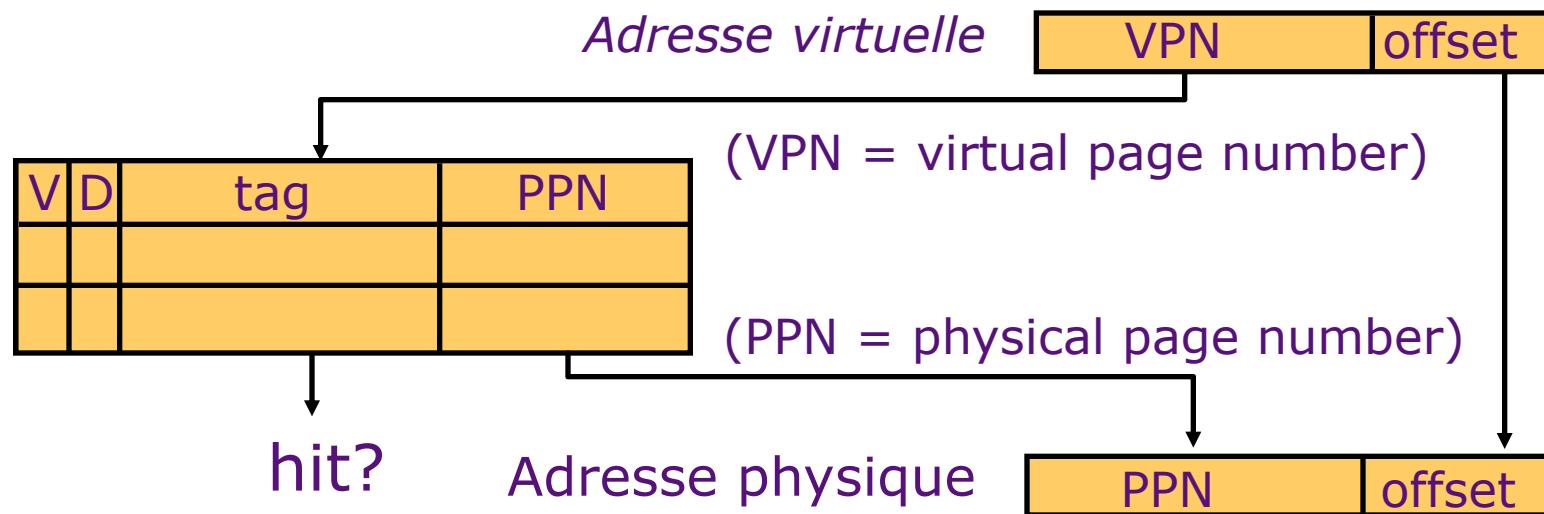
Solution: on utilise une sorte de cache dédié au translation, le TLBC

TLB hit

⇒ *Translation rapide en un cycle*

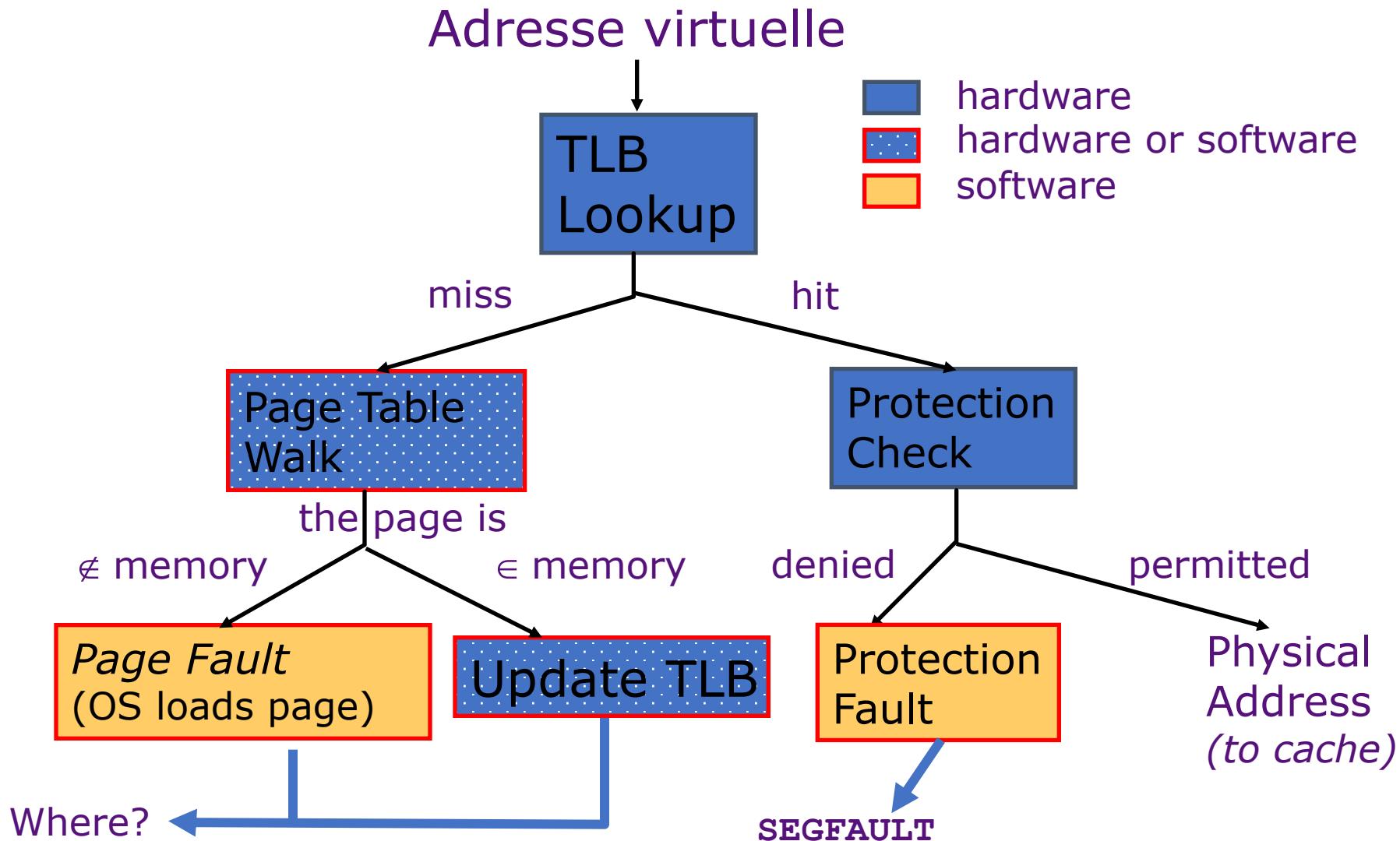
TLB miss

⇒ *accès à la table des pages et mise à jour du TLB*



Translation d'adresse:

résumé



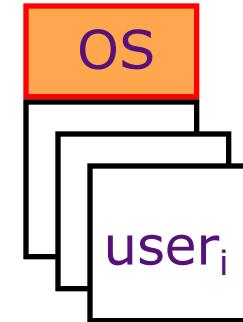
Les Systèmes à base de Mémoire Virtuelle

Protection & Privacy

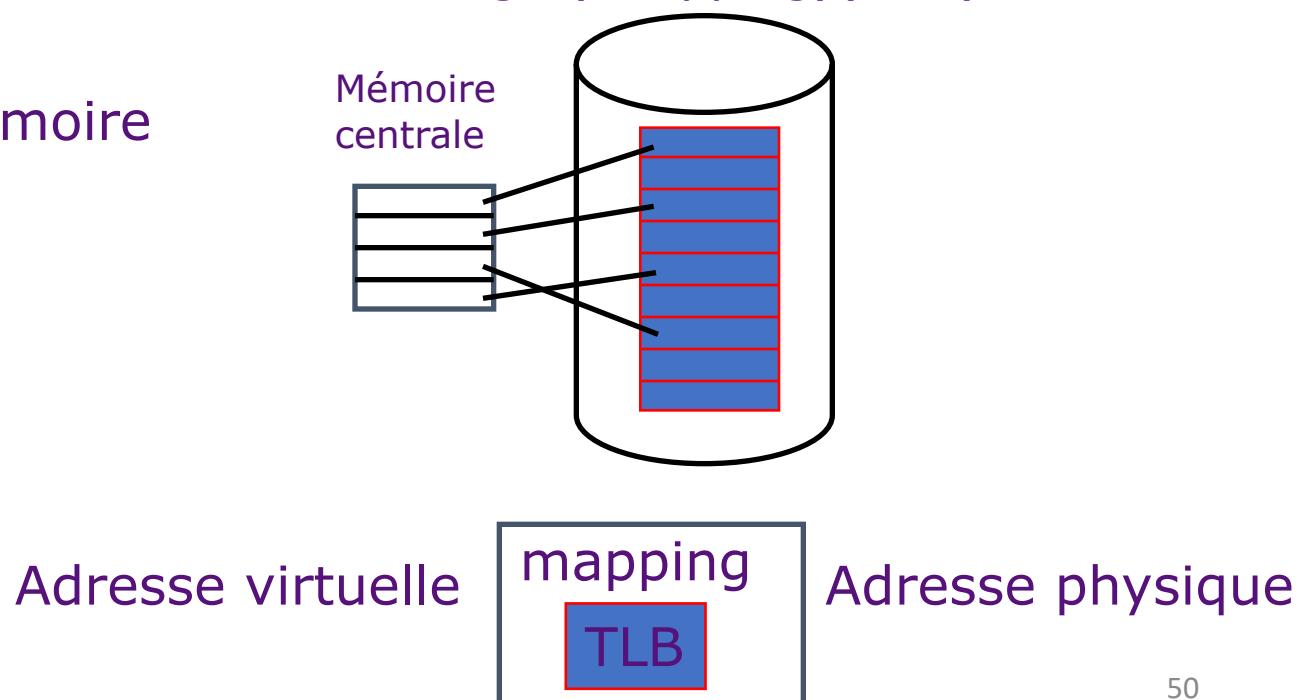
Plusieurs utilisateurs/processus, chacun possède son espace d'adressage privé.

Pagination

Donne la possibilité d'exécuter des programmes plus grands que la mémoire centrale.



Echange (swapping)(Disk)



Adresse virtuelle



Adresse physique

Pagination - Algorithme

