Exercice 1 :

1. La mémoire virtuelle est un mécanisme permettant au processeur de dissocier l'espace d'adressage utilisé par le processeur de la mémoire physique. Elle est basée sur le découpage de la mémoire en pages, et sur l'utilisation d'un mécanisme d'indirection faisant correspondre à un numéro de page virtuelle le numéro de page physique associée. Parmi les nombreux avantages apportés par l'utilisation de la mémoire virtuelle, on peut citer (il y en a plus de trois) :

{ dans un système multi-processus, le fait de pouvoir augmenter la quantité de mémoire physique allouée à un programme sans avoir à déplacer les pages des autres programmes en mémoire physique, la mémoire virtuelle permettant de donner l'illusion d'un espace d'adressage continu même si les pages physiques attribuées à un processus sont disséminées dans la mémoire physique ;

{ la protection contre les erreurs des autres programmes, chaque processus disposant de son propre espace d'adressage ;

{ la possibilité de disposer d'un espace d'adressage supérieur à la quantité de mémoire physique effectivement disponible sur le système, en couplant la mémoire virtuelle avec un mécanisme de va-et-vient ((swap)) ;

{ le fait pour le programmeur de ne plus avoir à se soucier de la quantité de mémoire physique disponible sur le système, la mémoire virtuelle étant gérée dynamiquement, à la différence de ce qui se fait avec les overlays ;

{ le partage de pages entre processus, chaque processus pouvant référencer les pages partagées, éventuellement à une adresse virtuelle différente, à travers sa table des pages.

2. Sur les systèmes à mémoire paginée, la MMU est un circuit matériel s'intercalant entre le processeur et le bus mémoire (il est d'ailleurs le plus souvent intégré sur la puce du processeur elle-même), et charge de convertir les adresses virtuelles manipulées par le processeur en adresses physiques. Au-delà de limiter la taille de la table de correspondance entre numéros de pages virtuelles et numéros de pages physiques, cette table peut être organisée de façon hiérarchique (c'est-à-dire arborescente), au point que

seules les portions utilisées de la table des pages soient effectivement présentes en mémoire centrale. Du fait de la grande taille et/ou de la structure hiérarchique de la table des pages, chaque demande d'opération mémoire peut se traduire par une ou plusieurs opérations de lecture en mémoire centrale, ce qui peut être très coûteux. Le rôle du TLB (Translation Lookaside Buffer) est de limiter ces accès coûteux aux tables des pages, en conservant dans une petite mémoire cache auxiliaire les correspondances entre numéros de pages virtuelles et physiques les plus récemment demandées. De par les principes de localité, la probabilité de réutiliser les correspondances déjà mémorisées est très grande, ce qui rend le TLB très efficace.

3. Les tailles des pages sont toujours des puissances de deux parce que cette écriture permet naturellement d'obtenir, à partir d'une adresse virtuelle donnée, le numéro de page et le déplacement (offset) dans la page : les bits de poids fort donnent le numéro de page, et les bits de poids faible le déplacement. Toute autre valeur de taille de page nécessiterait le calcul d'une division entière, bien plus coûteuse.
4. Le nombre de pages virtuelles du processus n'est pas important ; seul compte le fait que l'ensemble des pages du processus tient en mémoire physique. On veut connaître la fraction des accès mémoire résolus grâce à la TLB. c'est-à-dire le taux tel que : $3a + 10(1 - a) = 5$; $a = 71,4 \%$.
5. Le nombre d'entrées dans la table des pages est le nombre de pages présentes dans l'espace d'adressage virtuel, soit : $264/212 = 252$; c'est-à-dire plus de 4 millions de milliards d'entrées.
6. Plutôt que de stocker les correspondances entre pages virtuelles et pages physiques, ce qui implique d'avoir autant d'entrées que de pages virtuelles, et devient donc très important pour les gros processus, on maintient une table des pages inversée qui contient, pour chaque page physique, la page virtuelle associée. Comme les pages virtuelles sont propres à chaque processus, on doit stocker, en plus du numéro de page virtuelle, le numéro du processus possédant la page. L'avantage de cette technique est que la table des pages inversée est bien moins coûteuse en espace qu'une table des pages par processus. Cependant, pour chaque accès mémoire, il faut parcourir l'ensemble de la table pour savoir si la page virtuelle recherchée est présente ou non en mémoire physique, l'indice de la table auquel est trouvé le numéro de la page virtuelle représentant alors le numéro de la page physique associée. Sans optimisations, cette technique serait inexploitable. De même, alors que la table directe permet

directement de savoir si une page n'a jamais encore été allouée ou bien si elle a été swappée, avec la table inversée, si la page n'a pas été trouvée, il faut explorer le swap, par une technique similaire, pour savoir si la page s'y trouve. Ceci est cependant moins grave, car le temps de parcours des tables du swap est bien moins coûteux qu'un seul accès disque.

7. Un premier mécanisme accélérateur est ici encore la TLB. Grâce à elle, on n'entreprendra la recherche dans la table inverse que si le numéro de page virtuelle demandée ne fait pas partie des derniers numéros demandés et archivés dans le cache de la TLB. Pour cela, il serait trop coûteux d'imaginer d'implémenter une structure de recherche arborescente (de type b-arbre), car même un coût logarithmique par rapport au nombre de pages est trop important. Pour avoir un coût quasi-constant, on peut mettre en œuvre un mécanisme de table de hachage, où la clé est constituée du numéro de processus et du numéro de page virtuelle, et qui renvoie l'indice dans une table possédant le triplet (numéro de processus, numéro de page virtuelle, numéro de page physique). Si la correspondance est trouvée, on a le bon numéro de page ; sinon, on parcourt la table linéairement à partir de l'indice fourni par la clé jusqu'à trouver le bon triplet ou bien une place libre, qui indique que l'association n'est pas présente. En augmentant la probabilité de trouver rapidement la bonne clé, on peut utiliser une table des pages inverse de plus grande taille que le nombre réel de pages physiques ; elle sera toujours bien moins importante qu'une table des pages directe.
8. Avec la version simple de la table des pages inverse, du fait que l'association entre numéro de page virtuelle et numéro de page physique est globale, et qu'on n'a qu'une association possible d'un numéro de page virtuelle pour une page globale donnée, on ne peut associer une page physique à plusieurs pages virtuelles de différents processus, donc on ne peut mettre en œuvre de mécanisme de mémoire partagée. Pour rétablir ces associations, on peut mettre en œuvre le mécanisme de table de hachage décrit ci-dessus, en autorisant l'insertion de plusieurs triplets ayant le même numéro de page physique, mais un numéro de processus différent. Cependant, dans ce cas, on est obligé de limiter le nombre de pages partagées, afin de ne pas remplir complètement la table de pages inverse.

Exercice 2 :

1. La taille d'une entrée de la table des pages est de 4 octets (32 bits), donc la taille d'une page de la table des pages est de $210 \times 4 = 212$ octets, soit 4 ko. La hiérarchie des pages correspond à une page racine et à 210 pages de premier niveau indexées par la page racine, donc sa taille totale est de $212 \times (1 + 210) = 222 + 212$ octets, soit un peu plus de 4 Mo.

Il faut cependant remarquer que les adresses contenues dans les entrées des pages de la table des pages sont toutes des adresses physiques (en mémoire ou dans le va-et-vient). Ceci veut dire que, bien que la table des pages occupe des pages physiques dans la mémoire centrale au détriment du processus, ces pages n'apparaissent pas dans l'espace d'adressage virtuel du processus. L'espace d'adressage virtuel utile de chaque processus (et du système) est donc de 232 octets, quel que soit le nombre de pages physiques occupées par la table des pages.

2. La fraction de l'espace mémoire physique d'un processus (c'est-à-dire le nombre total de pages physiques occupées par le processus, qui est équivalent à l'espace disque occupé lorsque le processus est swappé sur disque), varie selon la taille du processus, mais également selon la disposition des pages du processus allouées, qui conditionne le nombre de pages de premier niveau devant être allouées. Le ratio de valeur maximale est obtenu pour un processus constitué d'une unique page, pour lequel deux pages doivent être allouées dans la table des pages (une pour la page racine, et une pour la page de premier niveau indexant la page du processus). On obtient alors un ratio de $2/1 = 2$. Une page de premier niveau sert complètement lorsque toutes ses entrées servent à indexer des pages de l'espace d'adressage du processus. De même, la page racine sert complètement lorsque toutes ses entrées sont utilisées. Le ratio de valeur minimale est donc obtenu lorsque l'ensemble des pages du processus est alloué. Il est alors de $(1+1024)/(1024 \times 1024) = 1025/1048576 \geq 1/1024$.