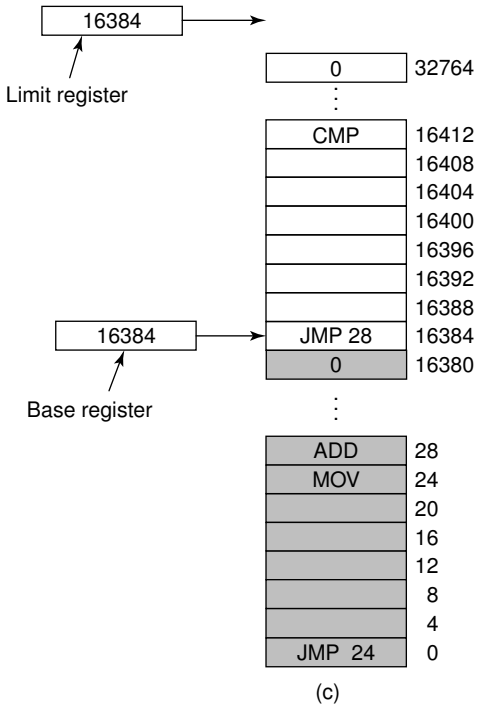
EX03 – Gestion de la mémoire

Systèmes d'exploitation / Classes T-2adfg

– Solutions –

1. À la figure suivante, les registres de base et de limite contiennent la même valeur, 16 384. Est-ce un hasard ou est-ce toujours le cas ? Si c'est un hasard, à quoi cela est-il dû ?



Solution : C'est un hasard. Le registre de base contient 16 384 parce que le programme a été chargé à cette adresse-là. Il aurait pu être mis n'importe où ailleurs. Le registre de limite contient 16 384 parce que le programme fait 16 384 octets. Il aurait pu faire quelle autre taille.

2. Comparez l'espace de stockage nécessaire pour mémoriser la mémoire libre, d'une part avec une table de bits, d'autre part avec une liste chaînée. La mémoire de 128 Mo est allouée par unités de n octets. Pour la liste chaînée, nous supposons que la mémoire est constituée de segments de 64 Ko en alternance avec des zones vides de taille identique. Nous admettons également que chaque nœud de cette liste nécessite une adresse mémoire de 32 bits, une longueur de 16 bits et un champ de 16 bits pour le prochain nœud. Quelle est la taille de stockage nécessaire pour chaque méthode ? Laquelle est la meilleure ?

   Solution : Pour la table de bits, on a besoin de 1 bit par unité d'allocation. Avec $2^{27}/n$ unités d'allocations, cela représente $2^{24}/n$ octets. La liste chaînée contient $2^{27}/2^{16}$ ou 2^{11} nœuds, chacun de 8 octets pour un total de 2^{14} octets. Pour un n de petite taille, la liste chaînée est plus adaptée. Pour un n de grande taille, la table de bits est plus intéressante. On peut calculer le point d'équilibre en comparant ces deux formules et en les résolvant pour n. Le résultat est $1\,Ko$. Pour n inférieur à $1\,Ko$, la liste chaînée représente le meilleur choix. Pour n supérieur à $1\,Ko$, la table de bits est préférable. Il est entendu que l'alternance de segments et de vides tous les 64 Ko est irréaliste. En outre, il faut $n \leq 64\,Ko$ si les segments et les vides font $64\,Ko$.

3. Considérons un système de va-et-vient dans lequel la mémoire est constituée par une succession de zones vides dans l'ordre suivant : 10 Ko, 4 Ko, 20 Ko, 18 Ko, 7 Ko, 9 Ko, 12 Ko et 15 Ko. Quelle zone sera prise pour les requêtes de segments successives suivantes :

   a) 12 Ko

   b) 10 Ko

   c) 9 Ko

   pour la première zone libre (*first fit*) ? Répondez également à cette question pour le meilleur ajustement (*best fit*), le plus grand résidu (*worst fit*) et la zone suivante (*next fit*)

   Solution : La première section libre (*first fit*) prend $20\,Ko$, $10\,Ko$ et $18\,Ko$. Le meilleur ajustement (*best fit*) prend $12\,Ko$, $10\,Ko$ et $9\,Ko$. Le plus grand résidu (*worst fit*) prend $20\,Ko$, $18\,Ko$ et $15\,Ko$. La section suivante (*next fit*) prend $20\,Ko$, $18\,Ko$ et $9\,Ko$.

4. Une machine possède un espace d'adressage de 32 bits et des pages de 8 Ko. La table des pages, entièrement matérielle, est constituée d'un mot de 32 bits par entrée. Lorsqu'un processus démarre, la table des pages est copiée en mémoire principale, à raison d'un mot toutes les 100 ns. Si chaque processus s'exécute pendant 100 ms (incluant le temps de chargement de la table), quelle fraction du temps UC est réservée au chargement des tables des pages ?

   Solution : La table de pages contient $2^{32}/2^{13}$ entrées, soit $524\,288$. Il faut $52\,ms$ pour charger la table des pages. Si un processus obtient $100\,ms$, elles se répartissent en $52\,ms$ pour le chargement de la table des pages et $48\,ms$ d'exécution. On en déduit que 52% du temps est consacré au chargement des tables de pages.

5. Une machine a des adresses virtuelles sur 48 bits et des adresses physiques sur 32 bits.

   (a) Si les pages font 4 Ko, combien d'entrées y a-t-il dans la table des pages pour une pagination à un niveau ? Expliquez.

   Solution : On a besoin d'une entrée par page, soit $2^{36} = 64 \times 1\,024 \times 1\,024 \times 1\,024$ entrées, puisqu'il y a $36 = 48 - 12$ bits dans le champ numéro de page.

   (b) La machine a maintenant un TLB de 32 entrées. Imaginons un programme dont les instructions tiennent sur une page et qui lit séquentiellement des longs entiers dans un tableau qui s'étend sur des milliers de pages. Le TLB est-il alors un mécanisme efficace ?

   Solution : Pour les instructions, pas de problème. Pour les données on aura un taux de réussite de 100% jusqu'à ce que le programme passe à la page de données suivante. Puisqu'une page de $4\,Ko$ contient $1\,024$ entiers longs, il y aura un échec de TLB et un accès mémoire supplémentaire toutes les $1\,024$ références aux données.

6. Une machine a des adresses virtuelles sur 38 bits et des adresses physiques sur 32 bits.

   (a) Quel est l'intérêt principal d'avoir une table des pages multi-niveaux ?

   Solution : Une table de pages multi-niveau, en raison de sa structure hiérarchique, réduit le nombre de pages de la table des pages qu'on a besoin d'avoir en mémoire. En fait, dans un programme qui contient un grand nombre d'instructions et une bonne localité on n'a besoin que de la table des pages de haut niveau (une page), d'une page d'instructions et d'une page de données.

   (b) Avec une table des pages à deux niveaux, des pages de 16 Ko et des entrées sur 4 octets, combien de bits faut-il au champ de la table des pages de plus haut niveau et combien pour le second niveau ?

   Solution : On alloue $12\,bits$ par champ. Le champ déplacement nécessite $14\,bits$ puisqu'il faut adresser $16\,Ko$. Cela laisse $24\,bits$ pour les champs de la page. Puisque chaque entrée fait $4\,octets$, une page peut contenir 2^{12} entrées de table de pages ce qui implique $12\,bits$ pour indexer une page. En allouant $12\,bits$ à chacun des champs de page, on pourra adresser les 2^{38} Octets.

7. Un ordinateur ayant des adresses 32 bits utilise une table des pages à deux niveaux. Les adresses virtuelles ont trois parties : un champ de 9 bits qui représente le premier niveau, un champ de 11 bits qui représente le second niveau, et un déplacement. De quelle taille sont les pages ? Combien en existe-t-il dans l'espace d'adressage ?

   Solution : On utilise $20\,bits$ pour les numéros des pages virtuelles, laissant $12\,bits$ pour le déplacement, ce qui produit une page de $4\,Ko$. Avec $20\,bits$ par page virtuelle, on obtient 2^{20} pages.

8. Supposons qu'une adresse virtuelle de 32 bits soit divisée en 4 champs : a, b, c et d. Les trois premiers sont utilisés pour une table des pages à trois niveaux. Le quatrième champ, d, est le déplacement (offset). Le nombre de pages dépend-il de la taille des 4 champs ?

   Solution : Le nombre de pages dépend du nombre total de bits dans *a*, *b* et *c* combinés. Leur répartition dans les champs n'a aucune importance.

9. Un ordinateur dont les processus ont 1 024 pages dans leur espace d'adressage conserve ses tables des pages en mémoire. Le délai pour lire un mot de la table des pages est de 5 ns. Afin de réduire celui-ci, l'ordinateur a une mémoire associative (TLB) qui contient 32 paires (page virtuelle, case physique) et dans laquelle une recherche prend 1 ns. Quel doit être le pourcentage de pages trouvées dans la mémoire associative pour réduire le temps moyen à 2 ns ?

   Solution : La durée d'instruction utile est $1h + 5(1-h)$ où h est le taux de succès. Si l'on compare cette formule à 2 et qu'on la résout pour h, on trouve que la valeur minimale de h est 0.75.

10. Une machine a des adresses virtuelles sur 48 bits et des adresses physiques sur 32 bits. Les pages sont de 8 Ko. Combien d'entrées sont nécessaires pour la table des pages ?

    Solution : Avec des pages de $8\,Ko$ et un espace d'adressage virtuel de $48\,bits$, le nombre de pages virtuelles est de $2^{48}/2^{13}$, soit 2^{35} (environ 34 milliards).

11. Un ordinateur avec des pages de 8 Ko, une mémoire principale de 256 Mo et un espace d'adressage virtuel de 64 Go se sert d'une table des pages inversée pour implanter sa mémoire virtuelle. De quelle taille doit être la table de hachage pour que la longueur moyenne de la chaîne de hachage soit plus petite quel 1 ? Nous supposons que la taille de la table de hachage est une puissance de 2.

    Solution : La mémoire principale contient $2^{28}/2^{13} = 32\,768$ pages. Une table de hachage de $32\,K$ aura une chaîne principale d'une longueur de 1. Pour descendre sous 1, on doit passer à la taille suivante, soit $65\,536$ entrées. En répartissant les $32\,768$ entrées sur les $65\,536$ connecteurs de la table, on obtient une longueur de chaîne de 0.5, ce qui assure une recherche rapide.

12. Si l'algorithme FIFO est utilisé avec 4 cases mémoire et 8 pages, combien de défauts de pages se produiront avec la chaîne de références 0 1 7 2 3 2 7 1 0 3, si les 4 cases sont initialement vides ? Refaites l'exercice avec l'algorithme LRU.

Solution : Les cases pour FIFO sont les suivantes :

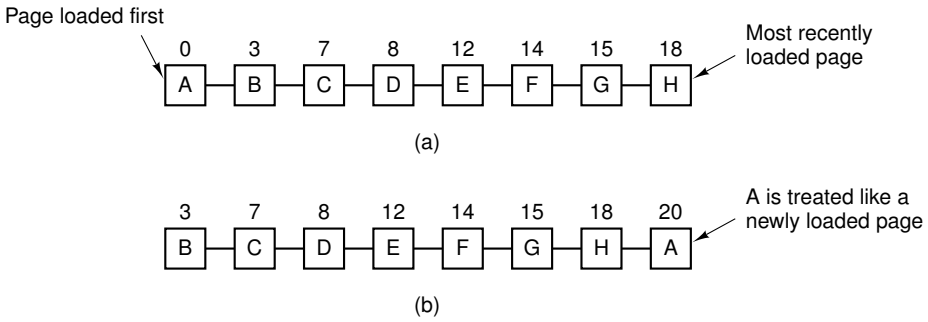
x	0	1	7	2	3	3	3	3	0	0
x	x	0	1	7	2	2	2	2	3	3
x	x	x	0	1	7	7	7	7	2	2
x	x	x	x	0	1	1	1	1	7	7

Les cases pour LRU sont les suivantes :

x	0	1	7	2	3	2	7	1	0	3
x	x	0	1	7	2	3	2	7	1	0
x	x	x	0	1	7	7	3	2	7	1
x	x	x	x	0	1	1	1	3	2	7

FIFO produit 6 défauts de page et LRU en produit 7.

13. Considérons la séquence de pages de la figure suivante (b). Supposons que les bits *R* des pages *B* jusque *A* sont respectivement 1 1 0 1 1 0 1 1. Quelles pages seront effacées avec l'algorithme de la deuxième chance ?



Solution : La première page contenant un bit 0 sera choisie, dans ce cas *D*.

14. Un petit ordinateur possède 4 cases mémoire. Au premier top d'horloge, les bits R sont égaux à 0 1 1 1 (0 pour la page 0 et 1 pour les autres). Aux tops suivants, les valeurs sont 1 0 1 1, 1 0 1 0, 1 1 0 1, 0 0 1 0, 1 0 1 0, 1 1 0 0 et 0 0 0 1. Si l'algorithme du vieillissement est utilisé avec un compteur 8 bits, donnez les valeurs des 4 compteurs après le dernier top d'horloge.

Solution : Les compteurs sont :

Page 0	0110110
Page 1	01001001
Page 2	00110111
Page 3	10001011

15. Donnez un exemple d'une suite de références à des pages qui conduisent à ce que la première page à remplacer soit différente suivant qu'on utilise LRU ou l'horloge. On suppose qu'on a alloué trois cases mémoire au processus et que la chaîne des références contient les numéros de pages 0, 1, 2 et 3.

Solution : La suite : 0, 1, 2, 1, 2, 0, 3. Avec LRU, la page 1 sera remplacée par la 3. Avec l'horloge, ce ne sera pas le cas puisque toutes les pages seront marquées et que le curseur sera sur la page 0.

16. Un ordinateur possède 4 cases. Nous donnons ci-après le moment du chargement, le temps du dernier accès et les bits *R* et *M* pour chaque page (les temps sont donnés en tops d'horloge) :

Page	Chargement	Dernière référence	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

(a) Quelle est la page qui sera remplacée avec l'algorithme NRU ?

Solution : Page 2

(b) Quelle est la page qui sera remplacée avec l'algorithme FIFO ?

Solution : Page 3

(c) Quelle est la page qui sera remplacée avec l'algorithme LRU ?

Solution : Page 1

(d) Quelle est la page qui sera remplacée avec l'algorithme de la deuxième chance ?

Solution : Page 2

17. Imaginez une situation dans laquelle il vaudrait mieux ne pas avoir de mémoire virtuelle plutôt qu'en avoir une. Expliquez.

Solution : Nul besoin de mémoire virtuelle lorsque les besoins mémoire sont bien identifiés et sous contrôle. Quelques exemples : processeurs spécialisés (comme les processeurs de réseau), processeurs embarqués, super-ordinateurs (pour la conception d'ailes d'avion, par exemple). On peut aussi préférer ajouter de la mémoire réelle. On peut également penser à simplifier le code. D'un autre côté, les idées issues de cette technique de mémoire virtuelle peuvent aussi être reprises comme on le fait, par exemple, avec les mémoires flash dans le cadre de la gestion des processus/threads.

18. Problème qui a inspiré une scène du film «The Social Network» :
«Suppose we are given a computer with a 16-bit virtual addresses, and a page size of 256 bytes. The system uses one-level page tables, which start at address `0x0400`. (The first few pages are reserved for hardware flags, etc. Maybe you wanted to have DMA on your 16-bit system, who knows?) Assume page table entries have eight status bits: 1 valid bit, 1 modify bit, 1 reference bit, and 5 permissions bits (this is a very secure system).

How many pages are there? How much memory do the page tables require?»

Solution : Avec un espace d'adressage sur 16 bits et des pages de 256 bytes, la table de pages comporte 256 entrées $(2^{16}/256 = 2^{16}/2^8 = 2^8 = 256)$. Notre système a donc 256 pages. Une entrée de la table a besoin de 8 bits pour l'état $(1+1+1+5)$ et de 8 bits pour le cadre de page, ce qui fait donc 16 bits, ou 2 Bytes. La table de pages fait donc $2 \times 256 = 512$ Bytes. Elle a besoin de 2 pages et occupe les adresses de `0x0400` à `0x05FF`.