Corrigé de l'exercice supplémentaire - Gestion de la mémoire

Enoncé

Dans un ordinateur à système paginé, la taille d'une page de 200 octets.
Soit le programme suivant:

```
Main()
{
   int A[20][20];
   int i,j;
   int B[20][20];
   for (i=0; i<20; i+5)
         for (j=0; j<20; j+19)
            B[i][j] =A[i][j];
   }
```

a. Sachant que les données sont distinguées du code, calculer le nombre de pages nécessaires au programme.
b. Donner la chaîne d'adresses produite par l'exécution programme, pour une organisation en lignes de toutes les matrices.
c. En déduire la chaîne de références correspondante.
d. Donner le nombre de défauts de pages causés pour une stratégie de remplacement LRU, et une mémoire centrale de 3 pages physiques.
e. On s'intéresse à expérimenter une nouvelle stratégie de remplacement, qui consiste à combiner LRU avec un bit de choix associé à chaque page. Ce bit est initialement à 0 pour toutes les pages. Dès qu'une page est sélectionnée comme page victime, son bit choix augmente de 1. L'algorithme choisit alors comme page victime la page la moins récemment utilisée ayant la plus faible valeur du bit de choix.
   Donner le nombre de défauts de pages. Y a-t-il une amélioration des résultats ? Discuter.

Solution

Taille page = 200 octets. Taille d'1 instruction = 4 octets
Pour déterminer le nombre de pages nécessaires au programme, il faut compter le nombre d'instructions du programme.
Le corps du programme comporte les instructions suivantes :

Programme	Equivalent	N°Instruction

Main()
{
int A[20][20];
int i,j;
int B[20][20];
for (i=0; i<20; i+5)
for (j=0; j<20; j+19)
B[i][j] =A[i][j];
}
	Début
	i=0;
	e1 : si i ≤ 19
	alors j=0;
	e2 : si j ≤ 19
	alors B[i, j]=A[i,j];
	j=j+19;
	goto e2;
	Fsi;
	i=i+4;
	goto e1;
	Fsi;
	Fin
		
		Inst 1
		Inst 2
		Inst 3
		Inst 4
		Inst 5
		Inst 6
		Inst 7
		Inst 8
		Inst 9
		Inst10
		Inst11

a. Nombre de pages nécessaires

Le programme est constitué de :

1. Code Exécutable : Pour déterminer le nombre de pages nécessaires au code, il faut calculer la taille du code.

Taille code = Nombre d'instructions x Taille d'1 instruction.
= 11x4 = 44 octets $\Rightarrow$ une page suffit pour le code.

Comme les données sont distinguées du code, le code sera stocké dans une page $\mathbf{P_0}$, isolée des pages de données (les variables i et j et les matrices A et B).

Remarque : la différence avec l'exercice traité en TD (programme des deux matrices) est que les variables *i* et *j* étaient dans la page de code $\mathbf{P_0}$; ce n'est pas le cas ici.

2. Données : D'après les déclarations, les données utilisées sont :

- Matrice A : A= 20x20 entiers = 400 entiers = 400x2 octets = 800 octets (sachant que 1 entier = 1mot = 2 octets) donc **A occupe une taille de 4 pages.**
- variable i : i = 1entier = 1mot = 2 octets.
- variable j : j = 1entier = 1mot = 2 octets.
- Matrice B : B = 20x20 entiers = 400 entiers = 400x2 octets = 800 octets.

$\Rightarrow$ Total données : 800+4+800 =1604 octets.

Taille d'une Page = 200 octets $\Rightarrow$ Nombre de pages nécessaires **aux données** = 1604/200= **9 pages**, numérotées de de $\mathbf{P_1}$ à $\mathbf{P_9}$.

Résumé : Le programme (code et données) nécessite 10 pages. On note ces pages :
$\mathbf{P_0, P_1, \ldots, P_9}$. On aurait pu les noter directement 0, 1, 2, …, 9

Représentation des pages virtuelles du programme

Pour représenter le contenu de ces pages, on adopte l'organisation en lignes des matrices A et B. Dans ce cas, chaque page d'une matrice contient exactement *5 lignes* (100 entiers).
Comme l'ordre des déclarations est A, i, j, puis B, le stockage des données va suivre cet ordre. D'où, la représentation des pages est comme suit :

$\mathbf{P_0}$: Code (programme)

$\mathbf{P_1}$: A [0, 0], ..., A[0, 19]
A [1, 0], …, A[1, 19]
A [2, 0]
A [3, 0]
A [4, 0], ….., A[4, 19]

$\mathbf{P_2}$: A [5, 0], ….., A[5, 19]
…..

A [9, 0], ….., A [9, 19]

$\mathbf{P_5}$: i, j,
B[0, 0] …..., B [0, 19]
B[1, 0],….., B [1, 19]

B[4, 0],….., B [4, 17]

$\mathbf{P_6}$: B[4, 18],….., B [4, 19]
B[5, 0] ,. …., B[5, 19]
……
B[9, 0], ….., B[9, 17]

P_3 : A [10, 0],…., A[10, 19]
…….
A [14, 0], …, A[14, 19]

P_4 : A [15, 0], …., A[15, 19]
……..
A [19, 0], …., A[19, 19]

P_7 : B[9, 18],….., B [9, 19]
B[10, 0],….., B [10, 19]
…….
B[14, 0],….., B [14, 17]

P_8 : B[14, 18],….., B [14, 19]
B[15, 0], ….., B [15, 19]
…….
B[19, 0],…… , B[19, 17]

P_9 : B[19, 18] , B [19, 19]

Remarque : les deux derniers éléments de B seont dans la page P9 à cause des variables ***i*** et ***j*** qui occupent deux emplacements de la page P5

b. Exécution détaillée du programme (références mémoires induites)

i=0	P0P5	//P0 contient l'instruction et P5 contient la variable i
Cond(i)	P0P5	//P0 contient l'instruction (i<=19) et P5 contient la variable i
j=0	P0P5	//P0 contient l'instruction et P5 contient la variable j
Cond(j)	P0P5	//P0 contient l'instruction (j<=19) et P5 contient la variable j
B[0,0] = A[0,0]	P0P1P5	//P0 contient l'instruction (affectation), P1 contient l'élément A[0,0] et P5 contient les variables i et j et l'élément B[0,0]
j= j+19	P0P5	
cond(j)	P0P5	
B[0,19] = A[0,19]	P0P1P5	
j= j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=4)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[4,0] = A[4,0]	P0P1P5	
j = j+19	P0P5	
cond(j)	P0P5	
B[4,19] = A[4,19]	P0P1P6	
j = j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=8)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[8,0] = A[8,0]	P0P2P6	
j = j+19	P0P5	
Cond(j)	P0P5	
B[8,19] = A[8,19]	P0P2P6	
j = j+19	P0P5	
Cond(j)	P0P5	
i= i+4 (i=12)	P0P5	
Cond(i)	P0P5	
j=0	P0P5	
Cond(j)	P0P5	
B[12,0] = A[12,0]	P0P3P7	
j = j+19	P0P5	

Matrice A
A[0,0]…A[4,19] page P1
A[5,0]…A[9,19] page P2
A[10,0]…A[14,19] page P3
A[15,0]…A[19,19] page P4

Matrice B
B[0,0]…B[4,17] page P5
B[4,18]…B[9,17] page P6
B[9,18]…B[14,17] page P7
B[14,18]…B[19,17] page P8
B[19,18], B[19,19] page P9

Cond(j)	P0P5
B[12,19] = A[12,19]	P0P3P7
j = j+19	P0P5
Cond(j)	P0P5
i= i+4 (i=16)	P0P5
Cond(i)	P0P5
j=0	P0P5
Cond(j)	P0P5
B[16,0] = A[16,0]	P0P4P8
J= j+19	P0P5
Cond(j)	P0P5
B[16,19] = A[16,19]	P0P4P8
J= j+19	P0P5
Cond(j)	P0P5
i= i+4 (i=20)	P0P5
Cond(i)	P0P5

(i>19) donc "Arrêt du programme"

c. La chaine de référence

P0P5 P0P5 P0P5 P0P5 P0P1P5 P0P5P0P5 P0P1P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P1P5 P0P5 P0P5 P0P1P6 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P2P6 P0P5 P0P5 P0P2P6 P0P5P0P5 P0P5 P0P5 P0P5 P0P5 P0P3P7 P0P5 P0P5 P0P3P7 P0P5 P0P5 P0P5 P0P5 P0P5 P0P5 P0P4P8 P0P5 P0P5 P0P4P8 P0P5 P0P5 P0P5 P0P5.

La chaine réduite est :

P0P5 P0P1P5 P0P5 P0P1P5 P0P5 P0P1P5 P0P5 P0P1P6 P0P5 P0P2P6 P0P5 P0P2P6 P0P5 P0P3P7 P0P5P0P3P7 P0P5 P0P4P8 P0P5 P0P4P8 P0P5.

On peut encore réduire la chaine en éliminant les séquences répétitives **(sachant qu'on a 3 frames libres au minimum)**

La chaine de référence réduite est donc:

P0P5 P0P1P5 P0P5 P0P1P6 P0P5 P0P2P6 P0P5 P0P3P7 P0P5 P0P4P8 P0P5

d. L'algorithme de remplacement LRU avec 3 frames

	P0	P5	P0	P1	P5	P0	P5	P0	P1	P6	P0	P5	P0	P2	P6	P0	P5	P0	P3	P7
F1	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0
F2		P5	P5	P5	P5	P5	P5	P5	P5	P6	P6	P6	P6	P2	P2	P2	P5	P5	P5	P7
F3				P1	P1	P1	P1	P1	P1	P1	P1	P5	P5	P5	P6	P6	P6	P6	P3	P3
Dfpage	D	D		D						D		D		D	D		D		D	D

	P0	P5	P0	P4	P8	P0	P5
F1	P0	P0	P0	P0	P0	P0	P0
F2	P7	P7	P7	P4	P4	P4	P5
F3	P3	P5	P5	P5	P8	P8	P8
Df_page		D		D	D		D

Taux de Défaut de pages = 14/27

e. Autre algorithme (LRU avec bit de choix)

Nouvelle stratégie : consiste à combiner LRU avec un **bit de choix** associé à chaque page. Ce bit est initialement à 0 pour toutes les pages. Dès qu'une page est sélectionnée comme page victime, son bit de choix augmente de 1. L'algorithme choisit alors comme page victime la page la moins récemment utilisée ayant la plus faible valeur du bit de choix.

L'algorithme de remplacement ***LRU avec bit* de choix (**MC sur 3 frames)

Remarque : Le bit de choix de chaque page est représenté dans la case en dessous du numéro de page

	P0	P5	P0	P1	P5	P0	P5	P0	P1	P6	P0	P5	P0	P2	P6	P0	P5	P0	P3	P7
F1	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P0	P6	P6	P6	P6	P3	P7
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
F2		P5	P5	P5	P5	P5	P5	P5	P5	P6	P6	P6	P6	P2	P2	P0	P0	P0	P0	P0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
F3				P1	P1	P1	P1	P1	P1	P1	P1	P5	P5	P5	P5	P5	P5	P5	P5	P5
				0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Dfpage	D	D		D						D		D		D	D	D			D	D

	P0	P5	P0	P4	P8	P0	P5
F1	P7	P7	P7	P4	P8	P8	P8
	0	0	0	0	0	0	0
F2	P0	P0	P0	P0	P0	P0	P0
	1	1	1	1	1	1	1
F3	P5	P5	P5	P5	P5	P5	P5
	1	1	1	1	1	1	1
Df_page				D	D		

Taux de Défaut de pages = 12/27

Conclusion : Il ya une amélioration par rapport à l'algorithme précédent car une page qui est déjà victime rentre de nouveau en mémoire avec une valeur de bit *incrémentée de 1* (évitera de la faire sortir une autre fois aussitôt). Donc, il s'agit d'augmenter les chances des pages (victimes) les plus fréquemment référencées pour rester le plus possible en mémoire centrale.