Conseil ⚠

Ça ne suffit pas de juste appliquer les formules va vous pénaliser sans comprendre ce qu'il se passe.

2018

Les détails suivants s'appliquent aux questions (b) - (d): Étant donné un système avec un espace d'adressage de 32 bits, une taille de page de 4 KB (2^{12} B) et avec la taille d'une entrée dans la table de pages de 4B:

(b) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(c) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page), combien de niveaux de pagination sont nécessaires et comment l'espace d'adressage est-il divisé ?

(d) (2 points) Considérant qu'un programme a besoin de 64 MB (2^{26} B) d'espace mémoire virtuel, combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (c).

2019

Question 1. *Accès mémoires (5 points au total)*

(a) (2 points) Considérons un système avec:
— Le temps d'accéder à la mémoire centrale est **200ns**
— Le temps pour faire un cherche dans le *Translation Lookaside Buffer* (TLB) est **20ns**
— Le temps pour traiter une *page fault* est **8000ns**
— Le taux de réussite du TLB est **90%**
— Le taux de *page fault* est **1 "manque (*miss*)" par 100** accès
Quel est le temps d'accès effectif (*effective access time* (EAT)) pour accéder à une page en mémoire ?

Les détails suivants s'appliquent aux questions (b) - (d): Étant donné un système avec un espace d'adressage de 64 bits, une taille de page de 1 MB et avec la taille d'une entrée dans la table de pages de 8B :

(b) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(c) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page), combien de niveaux de pagination sont nécessaires et comment l'espace d'adressage est-il divisé ?

(d) (1 points) Considérant qu'un programme a besoin de 1GB d'espace mémoire virtuel, combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (c).

Bonus (1 point) Dans le schéma d'adressage ci-dessus, pour un de les niveaux dans la hiérarchie, le nombre maximal de bits (pour que la table de pages puisse tenir en mémoire) n'est pas utilisé. Pourquoi est-il préférable que c'est le *premier niveau* de la hiérarchie qui utilise moins de bits ?

2020

Les détails suivants s'appliquent aux questions (c) - (f): Étant donné un système avec un espace d'adressage de 64 bits, une taille de page de 2MB et avec la taille d'une entrée dans la table de pages de 64b

(c) (1 point) Quelle est la taille (en *Bytes*) de la table de page à un niveau ?

(d) (1 point) Considérant que nous voulons que chaque table de pages dans un système de pagination hiérarchique s'inscrive dans un *frame* de mémoire (c'est-à-dire ne soit pas plus grande qu'une page),

i. (0.5 points) combien de niveaux de pagination sont nécessaires ?
ii. (0.5 points) combien de bits sont utilisés pour la table de pages la plus externe (*outer*) ?

(e) (1 points) Considérant qu'un processus avec 128MB stocké dans la memoire centrale, combien de rangée dans la table de page d'un niveau décrite en (c) vont être marquées "valide" ?

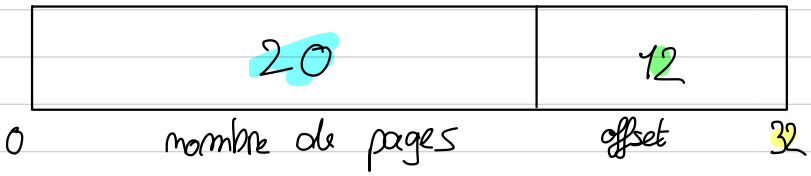
(f) (1 points) Considérant qu'un programme a besoin de 64GB d'espace mémoire virtuel, combien de tables de pages totales sont nécessaires pour la configuration hiérarchique décrite en (d).

Corrections

2018

b) $4KB = 2^{12}B$ $= 2^2 B$

$4B = 32\,b$ (taille entrée)

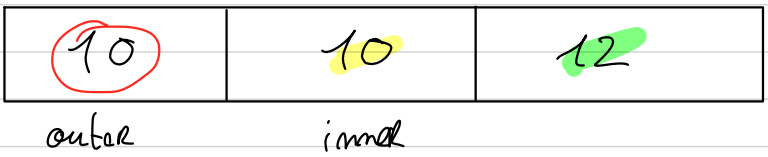


$$\frac{2^{32}}{2^{12}} \times 2^2 = 2^{22}B$$

c) $< 2^{12}B$ (taille d'une page)

$$\text{max} = \frac{2^{12}}{2^2} = 2^{10}B$$

Donc $20 - 10 - 10 - 0$ | 10 | 2 niveaux



d) $64MB = 2^{26}B$

On a besoin de $\frac{2^{26}}{2^{12}} = 2^{14}$ pages. $\rightarrow 10 < 14$ ∴

Nos tables sont de 2^{10} donc $\frac{2^{14}}{2^{10}} = 2^4$

$2^4 = 16$ inner tables

$+ 1$ outer pour accéder à ces tables $= 17$

$\Rightarrow 4 < 10$ donc 1 seul outer page suffit pour accéder les inner

2019

(b) $64b = 2^3 B = 8B$

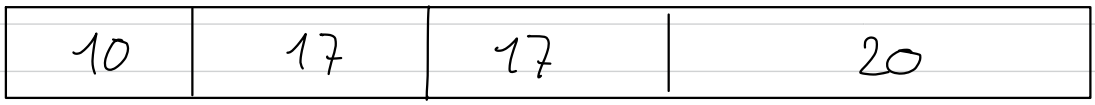
$1MB = 2^{20} B$

$$\frac{2^{64}}{2^{20}} \times 2^3 = 2^{47}$$

(c) 1 page $= 1MB = 2^{20} B$ $\quad max = \frac{2^{20}}{2^3} = 2^{17}$

44 | 17
3 | 2
10 |

Donc

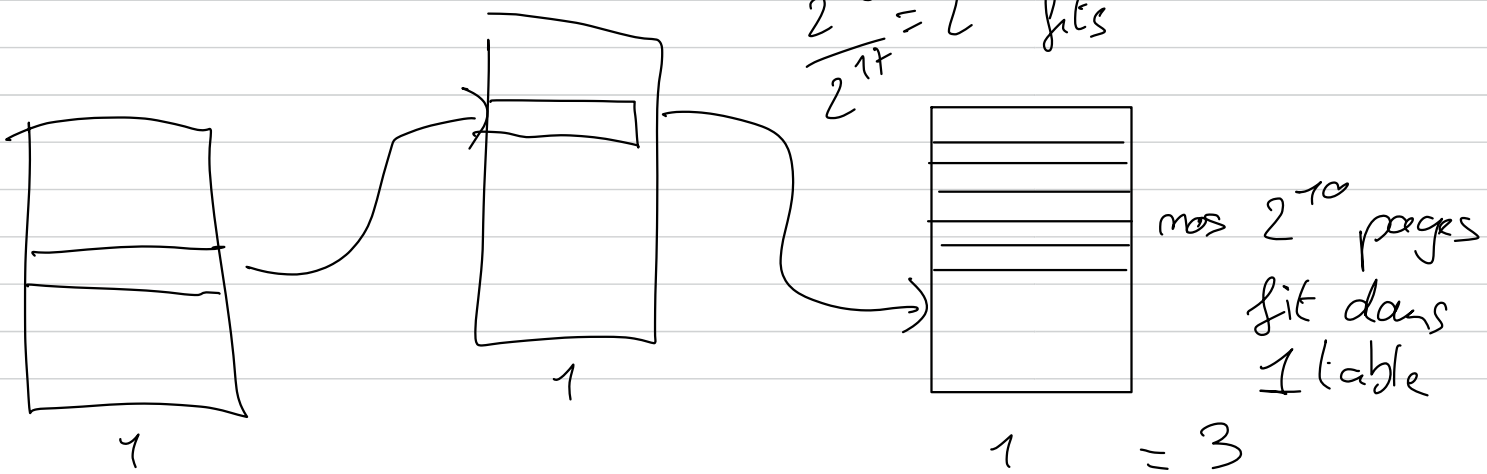


3 niveaux

(d) $16B = \frac{2^{30} B}{2^{20}} = 2^{10}$ pages

min 3 niveaux

$\frac{2^{10}}{2^{17}} = 2^{-7}$ fils



2020

(c) • espace d'adressage est de 64b

donc

• taille d'une page : $2\,MB = 2^{21}\,B$

$\frac{2^{64}}{2^{21}} = 2^{43}$ pages ← adresses possibles

• taille d'une entrée de table : $64b = 8B = 2^3 B$

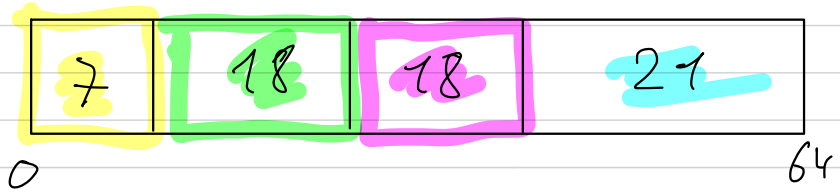
Donc $\frac{2^{64}}{2^{21}} \times 2^3 = 2^{46}\,B$ est la taille d'une table de page

(d) i. 1 page $= 2\,MB = 2^{21}\,B$

nb max par tabe : $\frac{2^{21}}{2^3} = 2^{18}$

43	48
⋮	2
7	

Donc



ii. Rappel sens : ←

+ outer + inner

Donc outer = 76

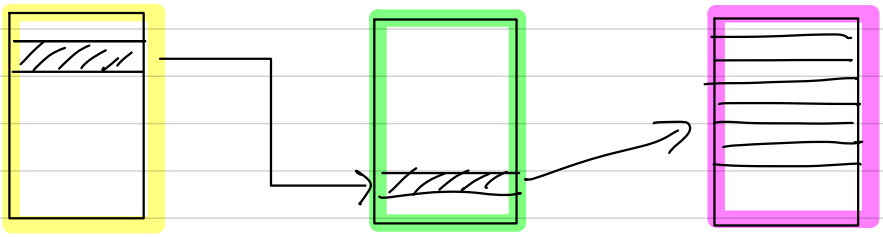
d) $128\,MB = 2^{27}\,B$

$$\frac{2^{27}}{2^{21}} = 2^{6} \text{ rangées}$$

e) $64\,GB = 2^{36}\,B$ min 3 niveaux

programme a besoin $\frac{2^{36}}{2^{21}} = 2^{15}$ pages

18 > 15 donc fils dans notre most inner page

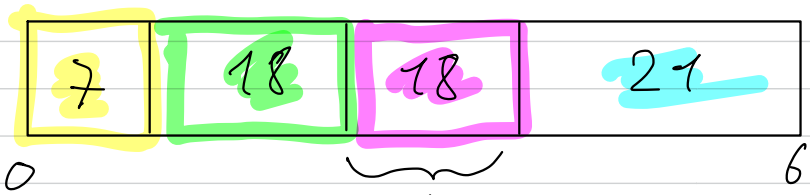


1 1 1 donc 3

Notre structure trouvé dans (d) :

taille de frame

nb rangé dans table de page



Combien d'espace notre most inner table peut accéder ?

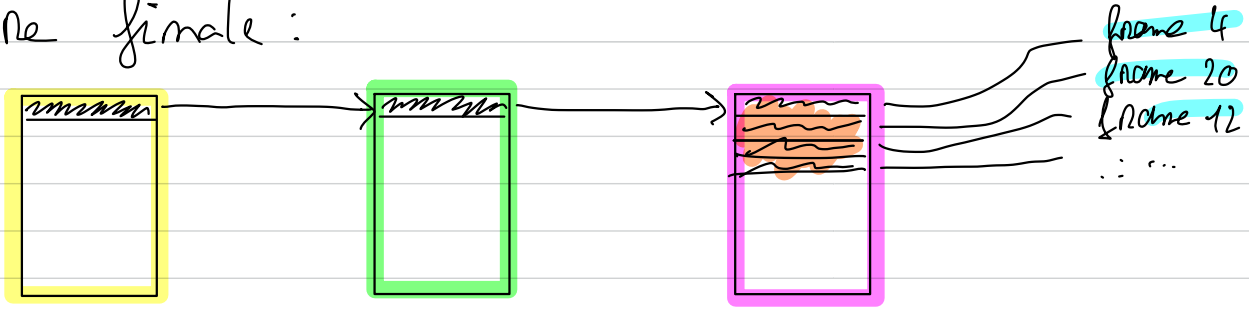
$$2^{18} \times 2^{21} = 2^{39} B$$

Le programme fais $64GB = 2^{36} B$

↳ $2^{39} B > 2^{36} B$ donc notre programme rentre dans notre table de page la plus inner.

Cependant on a minimum 3 table de pages.

Structure finale :



Entrainement :

Question 2. *Mémoire Virtuelle (4 points au total)*

Un programme s'exécute sur une machine et a été alloué **3 frames**. Il accède aux pages suivantes dans l'ordre indiqué:

1 4 3 2 4 5 1 2 5 6 2 5 3

En présumant que toutes les *frames* sont vides au départ (pagination a la demande pur), indique le contenu des *frames* et le nombre total de *page faults* pour chacun des algorithmes de remplacement de page suivants:

i. FIFO (*first in first out*)
ii. Deuxième Chance (*Second Chance* ou "horloge")
iii. Optimal
iv. "Least Recently Used"

Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, and seven frames? Remember that all frames are initially empty, so your first unique pages will cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

Corrections

Vu que notre stack est initialement vide et nos 3 première frame sont différentes, on aura min. toujours 3 page faults.

i) Remplace le plus vieux

1	4	3	2	4	5	1	2	5	6	2	5	3
1	1	1	2	2	2	2	2	2	6	6	6	3
	4	4	4	4	5	5	5	5	5	2	2	2
		3	3	3	3	1	1	1	1	1	5	5
Page faults :	1	1	1	1		1	1			1	1	1	1 = 10

Technique : quand on a un page fault, chercher le plus loins vers la gauche et elimine celui qui est dans le stack le plus longtemps

ii)

1	4	3	2	4	5	1	2	5	6	2	5	3
→1 1	→1 1	→1 1	2 1	2 1	→2 1	2 0	2 1	2 1	→2 0	→2 1	2 1	→2 0
	4 1	4 1	→4 0	→4 1	4 0	1 1	1 1	1 1	1 0	1 0	5 1	5 0
		3 1	3 0	3 0	5 1	→5 1	→5 1	→5 1	6 1	6 1	→6 1	3 1
	x	x	x	x		x	x		x		x	x = 9

Super video explicatif :

https://youtu.be/C26qsPwf-Js

iii) On prédit le future

1	4	3	2	4	5	1	2	5	6	2	5	3
1	1	1	1	1	1	1	1	1	6	6	6	wait
	4	4	4	4	5	5	5	5	5	5	5	
		3	2	2	2	2	2	2	2	2	2	
Page faults :												= 7

technique : remplace celui qui est le plus loin dans la queue

iv) Remplace le plus vieux

1	4	3	2	4	5	1	2	5	6	2	5	3
1_0	1_1	1_2	2_0	2_1	2_2	1_0	1_1	1_2	6_0	6_1	6_2	3_0
	4_0	4_1	4_2	4_0	4_1	4_2	2_0	2_1	2_2	2_0	2_1	2_2
		3_0	3_1	3_2	5_0	5_1	5_2	5_0	5_1	5_2	5_0	5_1
Page faults :												= 9

technique : on associe un age à chaque frame et à chaque nouvelle entrée de la queue, les tout le monde vieilli de +1. Puis on remplace le plus vieux.

Réponses exo livre

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7