TD2 : Gestion de la mémoire

RSA : Réseaux et Systèmes Avancés

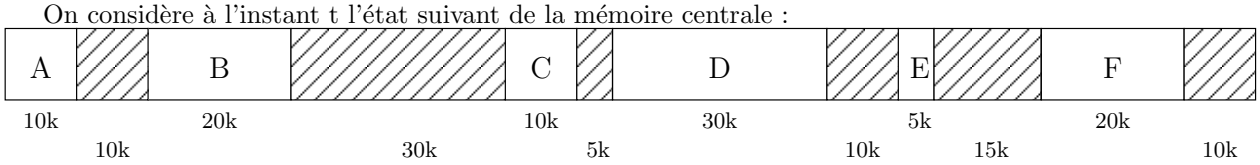
Deuxième année

★ **Exercice 1. Segmentation** (d'après Philippe Décogné).

Soit un système qui utilise l'allocation par partitions variables. On parle aussi d'allocation par zones quand la taille des zones allouées est variable, ou d'allocations par blocs quand la taille des blocs alloués est fixe.

L'allocation se fait selon le choix first fit. C'est-à-dire que la première zone rencontrée dont la taille est supérieure ou égale à la taille du processus à charger est celle qui est allouée au processus.

On considère à l'instant t l'état suivant de la mémoire centrale :



Représenter l'évolution de la mémoire centrale en fonction de l'arrivée des évènements successifs suivants.

▷ **Question 1.** Arrivée du programme G (20k)

▷ **Question 2.** Départ du programme B

▷ **Question 3.** Arrivée du programme H (15k)

▷ **Question 4.** Départ du programme E

▷ **Question 5.** Arrivée du programme I (40k)

★ **Exercice 2. Mécanismes de pagination** (d'après Philippe Décogné).

On considère une mémoire segmentée paginée pour laquelle les cases en mémoire centrale sont de 4 Ko. La mémoire centrale compte au total 15 cases numérotées de 1 à 15. Dans ce contexte, on considère deux processus A et B.

Le Processus A a un espace d'adressage composé de trois segments S1A, S2A et S3A qui sont respectivement de 8 Ko, 12 Ko et 4 Ko.

Le processus B a un espace d'adressage composé de deux segments S1B et S2B qui sont respectivement de 16 Ko et 8 Ko.

Pour le processus A, seules les pages 1 et 2 du segment S1A, la page 2 du segment S2A et la page 1 du segment S3A sont chargées en mémoire centrale respectivement dans les cases 4, 5, 10 et 6.

Pour le processus B, seules les pages 2 et 3 du segment S1B et la page 1 du segment S2B sont chargées en mémoire centrale respectivement dans les cases 11, 2 et 15.

▷ **Question 1.** Représentez sur un dessin les structures allouées (table des segments, tables des pages) et la mémoire centrale correspondant à l'allocation décrite.

▷ **Question 2.** Si 4098 et 12292 sont des adresses linéaires pour A, déterminez les adresses virtuelles et réelles correspondantes.

▷ **Question 3.** Même question avec 16389 pour A et 8212 pour B.

★ **Exercice 3. Taille de page optimale** (d'après Moufida Maimour).

Soit t la taille moyenne des processus et p (en octets) la taille d'une page. Soit e (en octets) la taille de l'entrée de chaque page dans la table des processus. Déterminer la taille optimale p d'une page ?

A.N. t = 64Ko et e = 8 octets

★ **Exercice 4. Intéret du TLB** (d'après Frédéric Vivien).

Un ordinateur dont les processus ont 1024 pages dans leur espace d'adressage garde toute sa table des pages en mémoire. Il faut 200 nano-secondes pour traverser cette table pour résoudre une adresse. Pour réduire ce temps, l'ordinateur possède une mémoire associative (TLB) qui contient 32 paires (page virtuelle, cadre). La recherche dans le TLB ne prend que 25 nano-secondes.

▷ **Question 1.** Quel doit être le pourcentage de pages trouvées dans le TLB (cache hits) pour que le temps de recherche moyen soit de 100 nano-secondes ?

▷ **Question 2.** Quel est le temps moyen pour un *cache hit* de 95% ?

★ **Exercice 5. Algorithmes de va-et-vient (swap) avec pagination** (d'après Cyril Rabat).

On considère une mémoire contenant 3 cadres de page et une mémoire virtuelle constituée de 5 pages (numérotées de 0 à 4). Les pages sont appelées comme suit : 0-L, 1-E, 2-L, 3-L, 4-E, 1-E, 2-L, 4-L, 0-E, 1-L (L pour lecture et E pour écriture). Donnez l'état des pages et des cadres pour chaque algorithme.

▷ **Question 1.** Algorithme FIFO

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1										
Cadre 2										
Cadre 3										
Défauts de page										

▷ **Question 2.** Algorithme optimal. Il décharge la page qui sera nécessaire le plus tard possible.

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1										
Cadre 2										
Cadre 3										
Défauts de page										

▷ **Question 3.** Algorithme LRU (Least Recently Used).

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1										
Cadre 2										
Cadre 3										
Défauts de page										

▷ **Question 4.** Algorithme de l'horloge (ie, de la seconde chance).

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1										
Cadre 2										
Cadre 3										
Défauts de page										
R de page 0										
R de page 1										
R de page 2										
R de page 3										
R de page 4										

▷ **Question 5.** Algorithme NRU (Not Recently Used). Il associe deux bits à chaque page. Leur valeur initiale est 0. M est mis à 1 quand la page est modifiée, et R est mis à 1 quand la page est référencée. Toutes les quatre instructions, les bits sont remis à zéro. L'ordre de préférence pour décharger une page est : M=R=0, M=1, R=1, M=R=1.

Page appelée	0-L	1-E	2-L	3-L	4-E	1-E	2-L	4-L	0-E	1-L
Cadre 1										
Cadre 2										
Cadre 3										
Défauts de page										
M et R de cadre 1										
M et R de cadre 2										
M et R de cadre 3										

★ **Exercice 6. Impact du workload** (d'après Moufida Maimour).

On considère un système à mémoire paginée où la taille d'une page est de `200*sizeof(int)`. Un petit processus occupe la page 0 de la mémoire. Il reste 3 cadres de pages libres à l'instant où un programme arrive. Ce dernier initialise le contenu du tableau A défini par `int A[][] = new int[100][100];`

Sachant que ce programme est chargé dans la cadre de page numéro 1 et que les 2 autres pages restantes sont initialement libres, quel est le nombre de défauts de pages expérimenté avec un remplacement LRU et ce pour les deux programmes d'initialisation suivants :

Algorithme 1

```
1 for (int j = 0; j < 100; j++)
2     for (int i = 0; i < 100; i++)
3         A[i][j] = 0;
```

Algorithme 2

```
1 for (int i = 0; i < 100; i++)
2     for (int j = 0; j < 100; j++)
3         A[i][j] = 0;
```