Contrôle continu - 23/10/2020

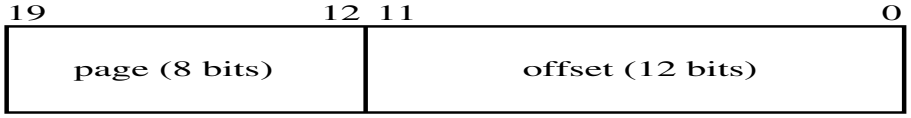
Durée : 1h30
Cours et/ou travaux dirigés autorisés.

Barème donné à titre indicatif : Ex.1 : 2 - Ex. 2 : 6 - Ex. 3 : 7 - Ex 4 : 5

Exercice 1.

1. On considère une architecture biprocesseur dotée d'un bus mémoire et d'un bus d'entrées-sorties. On suppose que l'on dispose sur cette architecture d'un noyau d'exécution pour gérer des processus. Préciser combien de processus au maximum peuvent être dans l'état actif à un instant donné sur une telle architecture.
2. Sur une architecture monoprocesseur, deux compilations sont lancées en parallèle. Expliquez pourquoi l'exécution *parallèle* de ces deux compilations durera moins longtemps que leur exécution séquentielle bien qu'il n'existe qu'un seul processeur central.
3. Par quelle stratégie peut-on éviter qu'un processus monopolise la ressource processeur pendant une durée trop longue ? Quel mécanisme matériel entrera en jeu pour mettre en œuvre une telle stratégie ?

Exercice 2. Supposons qu'une adresse en mémoire virtuelle paginée nécessite 20 bits organisés de la façon suivante :



1. Quelle est la taille de cette mémoire virtuelle, exprimée
   (a) en nombre d'octets et de pages ?
   (b) en Koctets ?
2. Quelle est l'adresse hexadécimale du $970^{\text{ème}}$ octet de la page 213 ?
3. À quel octet de quelle page correspond l'adresse hexadécimale $ABCDE$? Donner les résultats sous la forme ⟨n° page, n° octet⟩.

Exercice 3.

1. Segmentation
   On considère la table des segments suivante pour un processus $\mathtt{p}_1$:

Segment	Base	Limite
0	540	234
1	1254	128
2	54	328
3	2048	1024
4	976	200

   Calculer les adresses réelles correspondant aux adresses virtuelles suivantes (signaler éventuellement les erreurs d'adressage) : (0, 128), (2, 465), (3, 888).
2. Pagination
   Dans un système paginé, les pages font 256 mots mémoire et on autorise chaque processus

à utiliser au plus 4 cadres de la mémoire centrale. On considère la table des pages suivante du processus $\mathtt{p}_1$:

	Cadre	bit pres/abs
7	1 1 0	1
6	1 0 1	0
5	1 1 1	0
4	1 0 0	0
3	0 1 0	0
2	0 0 0	1
1	0 0 1	0
0	0 1 1	1

(a) Quelle est la taille exprimée en mots mémoire de l'espace d'adressage du processus $\mathtt{p}_1$?

(b) Calculer les adresses réelles (exprimées en décimal) correspondant aux adresses virtuelles suivantes (signaler éventuellement les erreurs d'adressage) : 240, 546, 1578.

(c) On considère l'adresse virtuelle suivante : 0000 0000 0000 0111. Sachant que les 4 bits de poids fort désignent le numéro de page et que les 12 bits suivants représentent le déplacement dans la page, donner l'adresse physique (exprimée en ⟨n° cadre, offset⟩) et l'adresse réelle (exprimée en binaire) correspondant à cette adresse.

3. Segmentation paginée
On considère un système avec une mémoire virtuelle segmentée paginée où la taille d'une page est de 4Ko et une mémoire physique de 64Ko. L'espace d'adressage d'un processus `p` est composé de trois segments S_0, S_1 et S_2 de tailles respectives 16Ko, 8Ko et 4Ko. À un moment donné, pour le processus `p`, les pages de numéros 1 et 2 du segment S_0, la page numéro 1 du segment S_1 et la page numéro 0 du segment S_2 sont chargées en mémoire physique, respectivement dans les cases 0, 2, 9, 12. Pour une donnée située dans l'espace d'adressage du processus `p` à l'adresse décimale 8212, indiquer :

(a) le segment

(b) le numéro de page dans le segment

(c) le déplacement dans la page

(d) le numéro de cadre

(e) le déplacement dans le cadre

(f) l'adresse physique (en décimal et en binaire).

Exercice 4. Algorithmes de remplacement de pages

1. Un programme possède un espace virtuel de 600 octets. L'unité adressable de la mémoire est l'octet. On considère la suite des adresses virtuelles suivante :

34, 123, 145, 510, 456, 345, 412, 10, 14, 12, 234, 336, 412.

Donner la suite des numéros de pages référencés, sachant qu'elles comportent 100 octets.

2. Le programme dispose de 300 octets en mémoire centrale. Montrer l'évolution de l'occupation des cadres en mémoire centrale et calculer le taux de défauts de page (en supposant la mémoire initialement vide) pour les algorithmes FIFO et LRU.