Université de M'sila
Faculté : MI Département :
Année Universitaire : 2022 / 2023
2ème Année Licence (2L)

Nom : ……
Prénom :……...……
Groupe : ……..……...

Examen de Systèmes d'Exploitation 1

Date : 24/05/2023 **Durée: 1h30 - Documentation non autorisée**

Exercice 1 : (Questions de Compréhension : 7 pts) (20 minutes)

Partie A) QCM : Mettez une croix sur une seule réponse (1 pt).

Q1) Parmi les commandes ci-dessous, laquelle permet d'afficher les processus en cours d'exécution ?

☐ dir ☒ ps ☐ man ☐ ls

Q2) Quelle commande permet d'interrompre un processus dans un système d'exploitation de type UNIX ?

☐ stop ☐ interrupt ☐ end ☒ kill

Q3) Un processus peut passer par les états (Actif, Prêt, Bloqué) dans l'ordre suivant :

☒ Actif -> Bloqué -> Prêt ☐ Bloqué -> Actif -> Prêt ☐ Actif -> Prêt -> Bloqué

Q4) Lors de l'initialisation du système (boot) tous les processus sont créés par le mécanisme fork.

☐ Vrai. ☒ Faux.

Partie B) Qui suis-je ? (3 pts).

Q5) Je suis un appel système qui affiche le PID du processus père.	**getppid()**
Q6) Je suis un appel système qui permet de créer un processus	**fork()**
Q7) Je suis un processus qui s'est terminé mais son père n'a pas encore lu son code de retour.	**Processus Zombie**
Q8) Je suis la différence entre le temps de la première exécution et le temps d'entrée dans le système.	**temps d'attente (TA)**
Q9) Je suis la différence entre le temps de terminaison et le temps d'entrée dans le système.	**temps de réponse (séjour) (TR)**
Q10) Je suis le premier programme qui est lancé à la mise sous tension de l'ordinateur.	**BIOS**

Partie C) Questions de Cours (3 pts).

Q11) Quelle est la différence entre un processeur et un processus ?

Un processeur est un composant physique capable d'exécuter des instructions.

Un processus est un ensemble d'instructions en cours d'exécution. U

Un processeur permet l'exécution des processus.

Q12) Décrire brièvement ce qu'est un ordonnanceur.

Un ordonnanceur est un module du noyau d'un système d'exploitation. Il sert à répartir la charge du processeur afin d'optimiser l'exécution des processus en parallèle.

Q13) Ecrire le microprogramme formel correspondant à l'exécution des instructions Assembleur suivantes (**Annexe** : Schéma d'un ordinateur sous le modèle de Von Neumann):

SUB @ ; // Faire ACC ← ACC – [@]	JMP @ ; // Effectue un saut à l'adresse @
Réponse :	**Réponse :**
H0 : CO // bus d'@ → RAM ;	H0 : CO // bus d'@ → RAM ;
H1 : Lecture, CO ++ ;	H1 : Lecture, CO ++ ;
H2 : RIM // bus de donnée → RI ;	H2 : RIM // bus de donnée → RI ;
H3 : RI.op // bus d'@ → RAM;	H3 : RI.op // bus d'@ → CO;
H4 : Lecture ;	
H5 : RIM // bus de donnée → RTURL ;	
H6 : ACC - RTURL → ACC ;	

Exercice 2 : (Edition de liens : 2 pts) (10 minutes)

La translation d'un module consiste à modifier son contenu pour qu'il puisse s'exécuter à un endroit diffèrent de celui pour lequel il était prévu initialement.

Q1) Donnez le résultat de la translation du module suivant en assembleur 68000, Si l'éditeur de liens décide de mettre la section de code à l'adresse hexadécimale **032340** et la section des données à l'adresse hexadécimale **043320**, les adresses génères sont sur **4 octets**.

Section code :					
AJOUT10 :	*move.w #10, D0*	*0*	*: 30 3C 00 0A*	**032340**	*: 30 3C 00 0A*
	jmp C	*4*	*: 4E F9 00 00 00 0E*	**032344**	*: 4E F9* ***00 03 23 4E***
AJOUT16 :	*move.w #16, D0*	*A*	*: 30 3C 00 10*	**03234A**	*: 30 3C 00 10*
C :	*add.w D1, D0*	*E*	*: D1 01*	**03234E**	*: D1 01*
	Move.w D0, MEMO	*10*	*: 33 C0 00 00 02*	**032350**	*: 33 C0* ***00 04 33 22***
	rts	*16*	*: 2E 75*	**032356**	*: 2E 75*
Section données :					
LOC :	*ds.w 1*	*0*	*: 00 00*	**043320**	*: 00 00*
MEMO :	*ds.w 1*	*2*	*: 00 00*	**043322**	*: 00 00*

Exercice 3 : (Ordonnancement : 4 pts) (20 minutes)

On considère un système possédant deux processeurs et une seule file d'attente pour les processus prêts.

Soit le scénario d'arrivée des processus suivants : A, B, C, et D, ayant les caractéristiques suivantes (la priorité 1 correspond à la plus faible priorité). Le CPU1 a la priorité d'accès à la file des processus prêts par rapport au CPU2.

Processus	Priorité	Instant d'arrivée	Durée d'exécution
A	2	0	4
B	4	2	5
C	3	0	6
D	1	0	7

Q1) Complétez les diagrammes, pour chacun des algorithmes de scheduling suivants :

a) FCFS (First Come First Served),

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
CPU 1	A	A	A	A	D	D	D	D	D	D	D															
CPU 2	C	C	C	C	C	C	B	B	B	B	B															
File d'attente	D	D	D	D	B	B																				
			B	B																						

b) Plus haute priorité préemptif (avec réquisition),

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
CPU 1	C	C	C	C	C	C	A	A																		
CPU 2	A	A	B	B	B	B	B	D	D	D	D	D	D	D												
File d'attente	D	D	A	A	A	A	D																			
			D	D	D	D																				

c) Round Robin (RR avec quantum = 2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
CPU 1	A	A	D	D	A	A	D	D	C	C	B															
CPU 2	C	C	B	B	C	C	B	B	D	D	D															
File d'attente	D	D	A	A	D	D	C	C	B	B																
			C	C	B	B																				

Q2) Donnez les temps d'attente (TA) et de réponse (TR) des processus.

	Algorithme FCFS		Algorithme Plus haute priorité		Algorithme RR avec Q = 2	
Processus	**Temps d'attente**	**Temps de Réponse**	**Temps d'attente**	**Temps de Réponse**	**Temps d'attente**	**Temps de Réponse**
A	0	4	4	8	2	6
B	4	9	0	5	4	9
C	0	6	0	6	4	10
D	4	11	7	14	4	11

Exercice 4 : (Gestion de la mémoire : 4 pts) (20 minutes)

On considère un système utilisant la technique de pagination et ayant les caractéristiques suivantes :

- Une table de page ayant **2^{16} entrées**
- Chaque entrée de la table de pages est codée sur **16 bits**. Une entrée contient un numéro de cadre de page et un bit de présence/absence.
- Le déplacement (offset) est codé sur **16 bits**
- Une adresse virtuelle indexe **2 octet**

Répondez aux questions suivantes en justifiant toujours votre réponse :

Q1) Quelle est la taille d'une page (un cadre)?

Taille d'une page = $2^{(offset)}$ * index @ virtuelle = 2^{16} * 2 octets = **128 Ko**

Q2) Quelle est la taille de la mémoire physique ?

Taille mem physique = nb de cadres * taille d'un cadre

= $2^{\text{nb de bits pour coder un cadre}}$ * taille d'un cadre

= $2^{(16-1)} * 2^{16} * 2 = 2^{32}$ octets = **4 Go**

Q3) Quelle est la taille de la mémoire virtuelle ?

Taille mémoire virtuelle = nb d'entrées TP * taille d'une page

= $2^{16} * 2^{16} * 2$ octets = 2^{33} = octets = **8 Go**

Q4) Quelle est la taille (en bit) du bus d'adresse de ce système ?

Taille d'un bus d'adresses = nb de bits pour coder la mémoire virtuelle = **33 bits**

Exercice 5 : (Algorithmes de remplacement de pages : 3 pts) (15 minutes)

Déterminez le nombre de défauts de page engendrés par les algorithmes FIFO, LRU et FIFO de la seconde chance sur la chaîne de références : 1, 2, 3, 1, 7, 4, 1, 2, 7, 4, 3, 1 avec 4 cadres de page :

FIFO : le nombre de défauts de page = 8												
La chaîne de références	**1**	**2**	**3**	**1**	**7**	**4**	**1**	**2**	**7**	**4**	**3**	**1**
Cadre 1	**1**	**1**	**1**	**1**	**1**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
Cadre 2		**2**	**2**	**2**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**1**
Cadre 3			**3**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**2**
Cadre 4					**7**	**7**	**7**	**7**	**7**	**7**	**3**	**3**
Défaut de page	**D**	**D**	**D**		**D**	**D**	**D**	**D**			**D**	

LRU : le nombre de défauts de page = 8												
La chaîne de références	**1**	**2**	**3**	**1**	**7**	**4**	**1**	**2**	**7**	**4**	**3**	**1**
Cadre 1	1	1	1	1	1	1	1	1	1	1	3	3
Cadre 2		2	2	2	2	4	4	4	4	4	4	4
Cadre 3			3	3	3	3	3	2	2	2	2	1
Cadre 4					7	7	7	7	7	7	7	7
Défaut de page	**D**	**D**	**D**		**D**	**D**		**D**			**D**	**D**

FIFO de la seconde chance : le nombre de défauts de page = 8												
La chaîne de références	**1**	**2**	**3**	**1**	**7**	**4**	**1**	**2**	**7**	**4**	**3**	**1**
Cadre 1	1^1	1^1	1^1	1^1	1^1	4^1	4^1	4^1	4^1	4^1	4^0	4^0
Cadre 2		2^1	2^1	2^1	2^1	2^0	1^1	1^1	1^1	1^1	1^0	1^1
Cadre 3			3^1	3^1	3^1	3^0	3^0	2^1	2^1	2^1	2^0	2^0
Cadre 4					7^1	7^0	7^0	7^0	7^1	7^1	3^1	3^1
Défaut de page	**D**	**D**	**D**		**D**	**D**	**D**	**D**			**D**	

<u>Exercice 6 :</u> (Bonus : 2 pts) (5 minutes)

Avec une taille moyenne de processus **P**, une taille de page **S** et une taille d'une entrée de la table de pages **E**, Quelle taille de page minimise l'espaces gaspiller en raison de la fragmentation interne et de la fragmentation de tables ?

Le nombre moyenne de page par processus est : **P / S**

L'espace nécessaire de la table de pages est : **P * E / S**

Le volume d'espace perdu en fragmentation interne est : **S / 2**

Donc l'espace total perdu est donné par l'équation : $\boldsymbol{Gaspillage} = \frac{P \times E}{S} + \frac{S}{2}$

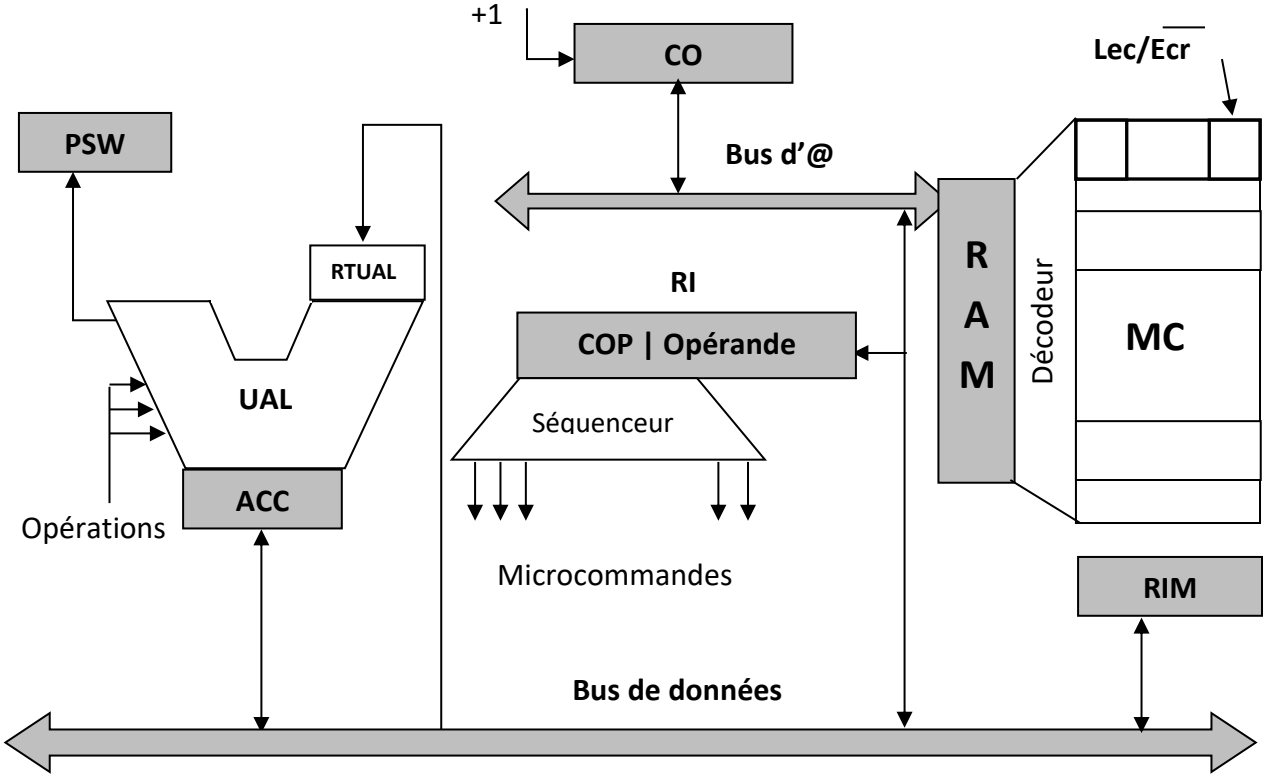
La valeur de S qui minimise le gaspillage est celle qui annule la dérivée première par rapport à S

$$-\frac{P \times E}{S^2} + \frac{1}{2} = 0$$

La résolution de l'équation obtenue donne : $S = \sqrt{2\,P\,E}$

Bon courage

ANNEXE


+1
CO
Lec/Ecr
PSW
Bus d'@
RTUAL
RI
R
A
M
Décodeur
MC
UAL
COP | Opérande
Séquenceur
ACC
Opérations
Microcommandes
RIM
Bus de données