GIF-1001 Ordinateurs: Structure et Applications
Hiver 2016
Examen final
26 avril 2016
Durée: 120 minutes

Cet examen comporte 7 questions sur 14 pages (incluant celle-ci), comptabilisées sur un total de 100 points. L'examen compte pour 40% de la note totale pour la session. Assurez-vous d'avoir toutes les pages. Les règles suivantes s'appliquent:

- Vous avez droit à une feuille aide-mémoire 8.5×11 recto-verso, écrite à la main, ainsi qu'une calculatrice acceptée.
- Écrivez vos réponses dans le cahier bleu qui vous a été remis;
- Nous vous fournissons deux annexes:
  - l'annexe A (p. 13) contient des rappels sur la conversion d'unités et sur les logarithmes;
  - l'annexe B (p. 14) contient une liste de conversion entre d'instructions ARM et de codes de conditions qui pourraient vous être utiles.

La table ci-dessous indique la distribution des points pour chaque question.

Question:	1	2	3	4	5	6	7	Total
Points:	10	10	10	20	20	20	10	100

Bonne chance et bon été!

1. (10 points) Un système communique avec un périphérique via une connexion série qui emploie le protocole RS-232, et qui possède les caractéristiques suivantes:

   - 8 bits par mot
   - parité impaire
   - 1 bit d'arrêt

   (a) (5 points) Ce système reçoit la chaîne de bits suivante: «01000010101». Quel est le mot reçu? Écrivez votre réponse en hexadécimal.

   Solution: Les bits sont envoyés du LSB au MSB. Les bits reçus sont:
   0 (start bit) - 1 - 0 - 0 - 0 - 0 - 1 - 0 - 1 - 0 (parité) - 1 (stop bit)
   Le mot reçu est donc 10100001, ou 0xA1.

   (b) (5 points) Ce même système reçoit la deuxième chaîne de bits suivante: «01101111001». Est-ce qu'il y a eu erreur durant la transmission? Si oui, quel bit est erroné? Détaillez votre démarche.

   Solution: Il y a eu erreur durant la transmission, parce que le nombre de «1» transmis est pair, donc le bit de parité aurait dû être à «1» (nous sommes en parité impaire). Cependant, il est à «0». Il est impossible de savoir quel bit est erroné.

2. (10 points) Les processus de la table 1 sont admis, dans l'ordre.

Nom du processus	durée (quanta)	priorité
P1	3	5 (basse)
P2	2	4
P3	3	3
P4	1	3
P5	2	1 (haute)

Table 1: Processus pour la question 2.

Pour les questions suivantes, indiquez quel processus sera exécuté à chaque quantum de temps pour l'algorithme spécifié. Si deux processus sont équivalents, commencez par celui ayant le chiffre le plus bas dans son nom.

(a) (1 point) Premier arrivé, premier servi.

Solution: P1–P1–P1–P2–P2–P3–P3–P3–P4–P5–P5

(b) (3 points) Plus court d'abord.

Solution: P4–P2–P2–P5–P5–P1–P1–P1–P3–P3–P3

(c) (3 points) Tourniquet.

Solution: P1–P2–P3–P4–P5–P1–P2–P3–P5–P1–P3

(d) (3 points) Priorité et tourniquet. Cet algorithme exécute le processus le plus prioritaire en premier. S'il y a plus qu'un processus ayant le même niveau de priorité, ceux-ci sont ordonnancés avec l'algorithme du tourniquet.

Solution: P5–P5–P3–P4–P3–P3–P2–P2–P1–P1–P1

3. (10 points) Un système utilise un contrôleur d'interruptions pour gérer les transferts entre les périphériques branchés sur ses bus et la mémoire principale. Le code assembleur ARM suivant illustre un court extrait du système d'exploitation (rudimentaire) qui gère ce système :

```
1   B       initialisation
2   NOP
3   NOP
4   NOP
5   NOP
6   NOP
7   NOP
8   B       traitePeripherique
9
10 initialisation
11   ; Operations
12   ...
13   B initialisation
14
15 traitePeripherique
16   ; Détermine l'adresse source (dans le périphérique)
17   ; Détermine l'adresse de destination (en RAM)
18
19   ; Démarre le transfert mémoire
20   BL copieMemoire
21
22   ; Fin du transfert
23   SUBS PC, LR, #4
```

Les questions ci-bas portent sur ce système.

(a) (2 points) À quoi correspondent les lignes 1 à 8 de ce code? À quoi servent-elles?

Solution: Elles correspondent à la table des vecteurs d'interruption. Elles servent à déterminer quoi faire lorsque chaque type d'interruption survient. Dans notre exemple, le système brancherait vers la fonction `traitePeripherique` lorsque survient une interruption FIQ.

(b) (3 points) Décrivez pourquoi il est avantageux d'utiliser les interruptions pour coordonner les transferts avec les entrées-sorties, plutôt que d'employer les entrées-sorties programmées.

Solution: Car le CPU peut faire autre chose en attendant que le périphérique soit prêt, ce qui n'est pas le cas des entrées-sorties programmées. C'est le périphérique qui indique quand il est prêt via une interruption.

(c) (5 points) Dans le système ci-haut, le CPU doit effectuer le transfert mémoire lui-même (ligne 20). Nommez une alternative qui permettrait au CPU de ne pas avoir à le faire, et décrivez brièvement son fonctionnement.

Solution: Les entrées-sorties par « Direct Memory Access » (DMA). Le DMA est un transfert de données direct entre un périphérique et la mémoire ou vice versa, effectué

sans intervention du microprocesseur. Cela est géré par un contrôleur de DMA, qui, lorsqu'il reçoit une requête, prend le contrôle des bus et pour synchroniser le transfert. Le microprocesseur peut faire autre chose pendant ce temps.

4. (20 points) Dans un système avec mémoire paginée où:

   - les pages ont une taille de 2Ko;
   - la taille de la mémoire virtuelle est de 256Mo;
   - la taille de la mémoire physique est de 32Mo;
   - un extrait de la table des pages (un «—» indique que la page n'est pas en RAM) est donné par :

Page	Trame
`0x00`	—
`0x01`	`0xB2`
`0x02`	`0xCD`
`0x03`	`0x05`
`0x04`	`0x9F`
`0x05`	`0x28`
`0x06`	—
`0x07`	`0x7C`
⋮	⋮

   (a) (4 points) Si la table des pages ne stocke que le numéro de trame pour chaque page, quelle est la taille totale de la table des pages? Écrivez votre réponse en kilo-octets (Ko).

   Solution: Le nombre total de pages est de $\frac{256\text{Mo}}{2\text{Ko}} = \frac{2^{28}}{2^{11}} = 2^{17}$.
   Le nombre total de trames est de $\frac{32\text{Mo}}{2\text{Ko}} = \frac{2^{25}}{2^{11}} = 2^{14}$. Nous avons donc besoin de 14 bits pour stocker le numéro d'une trame.
   La taille totale de la table des pages sera donc de $2^{17} \times 14\text{bits} = 224\text{Ko}$.

   (b) (4 points) Traduisez l'adresse virtuelle `0x2B35` en adresse physique en utilisant la table des pages ci-haut. Écrivez clairement votre démarche.

   Solution: L'adresse virtuelle `0x2B35` est séparée en deux: 1) les 11 LSB pour l'offset `0x335`, et 2) le reste pour le numéro de page `0x5`. Le numéro de trame correspondant à la page `0x5` est `0x28`, donc l'adresse résultante est `0x14335`.

   (c) Comme la mémoire virtuelle est plus grande que la mémoire physique, il est possible que le CPU veuille accéder à une page qui n'est pas en RAM. On dit alors qu'il y a faute de page.

      i. (4 points) Quelles sont les principales étapes à accomplir pour traiter une faute de page? Vous pouvez ignorer les caches et le mécanisme d'allocation sur le disque dur.

      Solution: Les étapes suivantes seront accomplies:

      1. Déterminer la trame à remplacer. Pour ce faire, l'algorithme LRU peut être utilisé.
      2. Écrire la page située dans la trame à remplacer sur le disque dur.
      3. Copier la page à charger du disque dur vers la RAM, dans la trame sélectionnée à l'étape 1.

4. Mettre à jour la table des pages (et autres structures de données comme la table des pages inverse).

ii. (4 points) Le CPU tente d'écrire à une adresse virtuelle en page `0x0`. Quel(s) est(seront) le(s) changement(s) à la table des pages après le traitement de la faute de page, si aucune trame n'est libre, et si la trame la moins récemment utilisée est la trame `0x9F`?

Solution: Les changements suivants seront apportés à la table des pages:
Page `0x00` : Trame `0x9F`
Page `0x04` : Trame —

(d) (4 points) Décrivez une façon simple d'implémenter l'algorithme LRU, «Least Recently Used». N'utilisez pas de code : ne faites que décrire son fonctionnement dans vos propres mots.

Solution: On peut, par exemple, utiliser un compteur pour chaque trame. Lorsque l'on accède à une trame, on incrémente les compteurs pour toutes les autres trames. Le compteur de la trame accédée est remis à zéro. La trame LRU est celle dont la valeur du compteur est la plus élevée.

5. (20 points) Un système possède les caractéristiques suivantes:

   - une seule cache (L1) de type «write-back»;
   - la cache stocke des blocs contenant 8 mots;
   - le temps d'accès à un mot en cache L1 est de 1ns;
   - la cache possède des blocs vides.

   Ce système est illustré dans la figure 1:

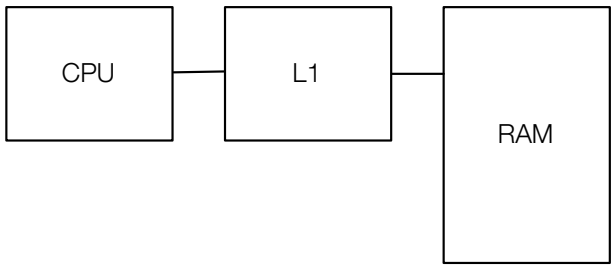

Figure 1: Système à une cache pour la question 5

Répondez aux questions suivantes portant sur ce système.

(a) (6 points) Décrivez les étapes nécessaires à la *lecture* du premier mot d'un bloc qui n'est *pas* présent en cache.

> **Solution:** Le bloc n'est pas présent en cache, nous avons donc un «miss». Il nous faut donc:
>
> 1. Trouver un bloc à remplacer en cache (par exemple grâce à l'algorithme LRU), on trouvera l'un des blocs vide;
> 2. Copier le bloc présent en RAM vers le bloc en cache;
> 3. Retourner le premier mot du bloc en question au CPU.

(b) (5 points) Décrivez les étapes nécessaires à l'*écriture* du deuxième mot présent dans le même bloc qu'en (a).

> **Solution:** Le bloc est présent en cache, nous avons donc un «hit». Il nous faut donc:
>
> 1. Écrire le mot dans le deuxième emplacement du bloc en cache;
> 2. Marquer le bloc comme «dirty».

Afin d'améliorer les performances du système, on décide de lui rajouter une deuxième cache (L2), située entre la cache existante (L1) et la mémoire principale (RAM), tel qu'illustré dans la figure 2.

(c) Sachant que le temps de transfert d'un bloc en RAM vers L2 prend 100ns et que le temps de transfert d'un bloc en L2 vers L1 prend 10ns, calculez le temps total d'accès aux 8 mots d'un même bloc lorsque:

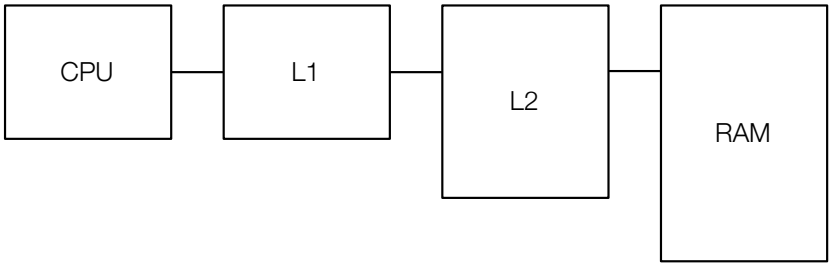

Figure 2: Système à deux caches pour la question 5.

i. (3 points) Le bloc est en cache L1;

Solution: Comme le bloc est en L1, on n'a qu'à lire les 8 mots directement en L1: $8 \times 1\text{ns} = 8\text{ns}$.

ii. (3 points) Le bloc est en cache L2;

Solution: Le bloc doit être préalablement transféré en cache L1 à partir de la cache L2. Cela prend 10ns. Ensuite, on lit les 8 mots directement en L1: $10\text{ns} + 8 \times 1\text{ns} = 18\text{ns}$.

iii. (3 points) Le bloc est en RAM.

Solution: Le bloc doit être préalablement: transféré de la RAM vers L2 (100ns), et de L2 vers L1 (10ns). Ensuite, on lit les 8 mots directement en L1: $100\text{ns} + 10\text{ns} + 8 \times 1\text{ns} = 118\text{ns}$.

6. (20 points) Considérons un système utilisant une allocation indexée dans son gestionnaire de fichiers. Ce système possède trois (3) fichiers: «bientot.txt», «enfin.txt», et «courage.txt». Les tables d'index pour chacun de ces fichiers ainsi que le contenu du disque dur sont illustrés dans la figure 3.

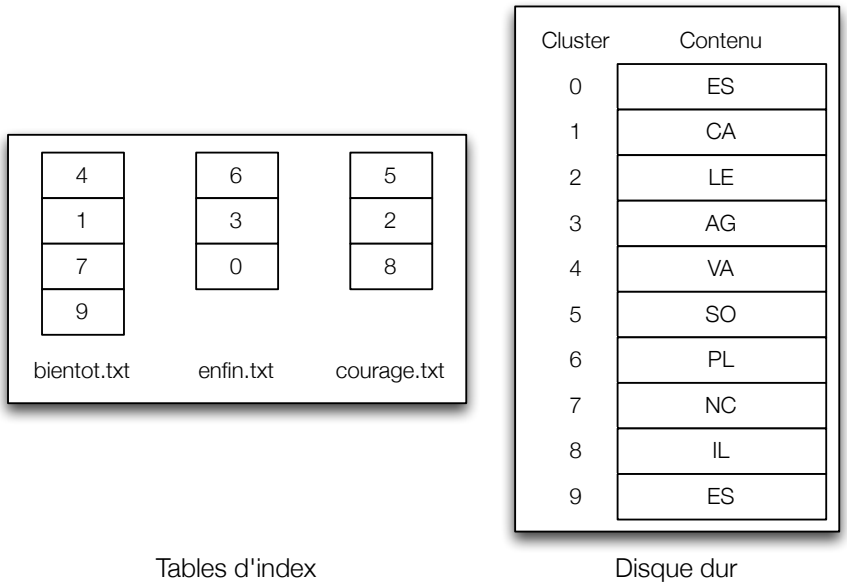

Figure 3: Tables d'index et contenu du disque dur pour les trois fichiers de la question 6.

(a) (3 points) Quel est le contenu de chacun des trois fichiers?

Solution: «bientot.txt»: VACANCES, «enfin.txt»: PLAGES, «courage.txt»: SOLEIL

(b) (7 points) Quelle(s) serai(en)t la(les) structure(s) de données nécessaire(s) pour représenter ces trois fichiers avec un système d'allocation chaînée avec table d'allocation (FAT)? Écrivez son(leur) contenu.

Solution:

Table des répertoires

Nom du fichier	Premier cluster
bientot.txt	4
enfin.txt	6
courage.txt	5

Table d'allocation

Cluster	Cluster suivant
0	~
1	7
2	8
3	0
4	1
5	2
6	3
7	9
8	~
9	~

(c) Considérons maintenant un système utilisant l'allocation FAT32. Sachant qu'un système FAT32 utilise 32 bits pour représenter les clusters :

i. (4 points) Calculez la taille d'un cluster si la taille du disque dur est de 512Go et qu'on utilise le nombre maximum de clusters supporté par FAT32. Vous pouvez assumer que les clusters divisent le disque dur en entier, et en parts égales.

Solution: 512Go $= 2^{39}$o. $\frac{2^{39}}{2^{32}} = 2^{39-32} = 2^7 = 128$ octets par cluster.

ii. (4 points) Calculez la taille de la table d'allocation (FAT) de ce système.

Solution: 32 bits = 4 octets. $2^{32} \times 4 = 2^{32}2^2 = 2^{32+2} = 2^{34} = 2^4 2^{30} = 16$Go.

iii. (2 points) Pourquoi les systèmes d'exploitation modernes n'utilisent-ils plus l'allocation chaînée avec table d'allocation (FAT)?

Solution: La taille de la FAT est très élevée, et contient l'information pour *tous* les fichiers. Il est beaucoup plus efficace de stocker de l'information pour chaque fichier séparément, comme dans l'allocation indexée.

7. (10 points) Répondez aux questions suivantes par vrai ou faux.

   (a) (1 point) Le TLB est une cache spéciale du MMU pour la table des pages.

   Solution: Vrai.

   (b) (1 point) Il est impossible d'avoir de la fragmentation externe en allocation contiguë avec partitions de taille variable.

   Solution: Faux.

   (c) (1 point) Dans un système d'exploitation moderne, c'est le programme qui doit redonner le contrôle au système.

   Solution: Faux.

   (d) (1 point) Le bus USB 2.0 simule les interruptions.

   Solution: Vrai.

   (e) (1 point) En communication série, le mode différentiel permet de détecter les erreurs de transmission.

   Solution: Faux.

   (f) (1 point) Il ne peut y avoir qu'un seul processus dans l'état «prêt» à la fois.

   Solution: Faux.

   (g) (1 point) Le BIOS sert à lancer le système d'exploitation.

   Solution: Vrai.

   (h) (1 point) Les modules d'entrée-sortie sont des interfaces entre les périphériques et le microprocesseur.

   Solution: Vrai.

   (i) (1 point) En ARM, une interruption logicielle et une interruption de type «instruction non-définie» ne peuvent survenir en même temps.

   Solution: Vrai.

   (j) (1 point) On utilise l'algorithme du tourniquet pour déterminer l'ordre de traitement des interruptions.

   Solution: Faux.

A Annexe: Unités et logarithmes

A.1 Unités

Petit rappel sur les unités:

1Ko	$= 2^{10}$		$=$	1 024 octets
1Mo	$= 2^{20}$	$= 1\,024$Ko	$=$	1 048 576 octets
1Go	$= 2^{30}$	$= 1\,024$Mo	$=$	1 073 741 824 octets

A.2 Logarithme en base 2

Il est facile de calculer des logarithmes en base 2 à partir de logarithmes dans une autre base N (ex: 10). Pour ce faire, appliquez l'équation suivante:

$$\log_2 x = \frac{\log_N x}{\log_N 2}\,.$$

B Annexe: Instructions ARM et codes de conditions

Instruction	Description
`ADD Rd, Rs, Op1`	`Rd` ← `Rs + Op1`
`AND Rd, Rs, Op1`	`Rd` ← `Rs AND Op1`
`ASR Rd, Rs, #imm`	`Rd` ← `Rs` / $2^{\texttt{imm}}$
`Bcc Offset`	`PC` ← `PC + Offset`, si `cc` est rencontré
`BLcc Offset`	Comme `B`, `LR` ← adresse de l'instruction suivante
`CMP Rs, Op1`	Change les drapeaux comme `Rs - Op1`
`LDR Rd, [Rs, Op2]`	`Rd` ← `Mem[Rs + Op2]`
`LDR Rd, [Rs], Op2`	`Rd` ← `Mem[Rs]`, `Rs` ← `Rs + Op2`
`LDR Rd, [Rs, Op2]!`	`Rs` ← `Rs + Op2`, `Rd` ← `Mem[Rs]`
`LSL Rd, Rs, #imm`	`Rd` ← `Rs` × $2^{\texttt{imm}}$
`MRS Rd, CPSR`	`Rd` ← `CPSR`
`MUL Rd, Rs, Op1`	`Rd` ← `Rs` × `Op1`
`MVN Rd, Op1`	`Rd` ← `!Op1` (inverse les bits)
`POP {Reg List}`	Met la liste de registres sur la pile
`PUSH {Reg List}`	Met la liste de registres sur la pile
`STR Rd, [Rs, Op2]`	`Mem[Rs + Op2]` ← `Rd`
`STR Rd, [Rs], Op2`	`Mem[Rs]` ← `Rd`, `Rs` ← `Rs + Op2`
`STR Rd, [Rs, Op2]!`	`Rs` ← `Rs + Op2`, `Mem[Rs]` ← `Rd`
`SUB Rd, Rs, Op1`	`Rd` ← `Rs - Op1`

Table 2: Instructions ARM. `Op1` dénote une opérande de type 1, et `Op2` une opérande de type 2.

Code	Condition	Code	Condition
`CS`	Retenue (carry)	`CC`	Pas de retenue
`EQ`	Égalité	`NE`	Inégalité
`VS`	Débordement	`VC`	Pas de débordement
`GT`	Plus grand	`LT`	Plus petit
`GE`	Plus grand ou égal	`LE`	Plus petit ou égal
`PL`	Positif	`MI`	Négatif

Table 3: Codes de condition.