GIF-1001 Ordinateurs: Structure et Applications
Hiver 2017
Examen final
25 avril 2017
Durée: 180 minutes
Professeur: Jean-François Lalonde

Cet examen comporte 8 questions sur 19 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 trois annexes:
  - l'annexe A (p. 17) contient des rappels sur la conversion d'unités et sur les logarithmes;
  - l'annexe B (p. 18) contient une liste d'instructions ARM ainsi que des codes de conditions;
  - l'annexe C (p. 19) contient la table ASCII.

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

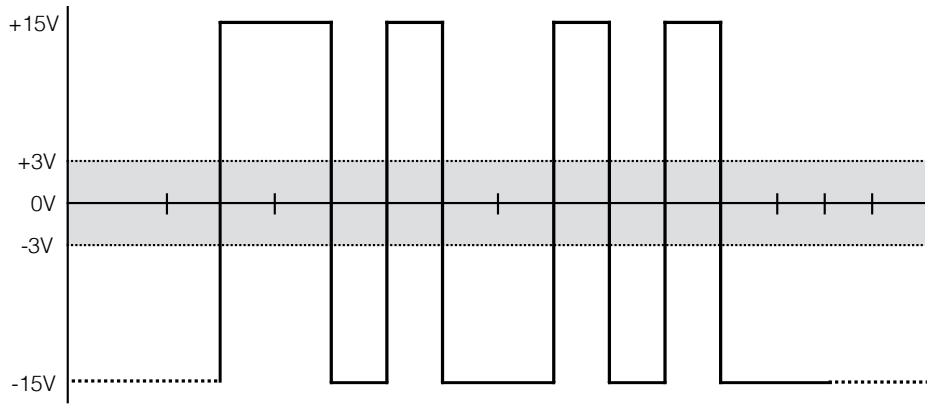
Question:	1	2	3	4	5	6	7	8	Total
Points:	12	10	15	15	8	15	12	13	100

Bonne chance et bon été!

1. (12 points) Un système communique avec un périphérique via une connexion série qui emploie le protocole RS-232, et qui possède les caractéristiques suivantes:

   - 7 bits par mot
   - parité paire
   - 2 bits d'arrêt

   (a) (5 points) Le système envoie le signal illustré dans la figure ci-bas. Sachant que la plage entre +3V et +15V corresponde à un «0» logique, et que la plage entre -3V et -15V corresponde à un «1» logique, quel est le mot envoyé? Écrivez votre réponse en hexadécimal.



Solution: Les bits du mot sont envoyés du LSB au MSB. Les bits reçus sont:
0 (start bit)–0–1–0–1–1–0–1–0 (parité)–1–1 (stop bits) (3 points pour les bits)
Le mot reçu est donc 1011010 (sur 7 bits), ou 0x5A. (2 points pour la bonne réponse)

   (b) Le système envoie ensuite la série de bits suivante sur la ligne: 0–1–0–1–1–0–1–0–0–1–1. De l'autre côté, le périphérique détermine qu'il a reçu le mot 0x39.

      i. (2 points) Y a-t-il eu erreur de transmission? Si oui, indiquez le mot que le périphérique aurait dû recevoir.

Solution: Oui (1 point), le périphérique aurait dû recevoir le mot 0x2D et non 0x39 (1 point).

      ii. (3 points) Est-ce que le périphérique peut savoir s'il y a eu erreur de transmission? Pourquoi?

Solution: Non (1 point), car le bit de parité est adéquat (2 points). Il ne peut détecter que si un nombre *impair* de bits a été modifié par le bruit.

On modifie la configuration du système pour qu'il envoie maintenant *8 bits par mot*. Les autres configurations demeurent inchangées.

   (c) (2 points) Si la vitesse de transfert est de 9600 bits par seconde, combien de temps ce système prendra-t-il pour transférer un fichier de 2Mo? Donnez votre réponse en minutes.

Solution: 2 Mo = 2^{20} o. Chaque octet est «emballé» avec 4 bits supplémentaires, donc nécessite 12 bits pour être envoyé.
Temps total: $\frac{(2\times 2^{20})\times 12}{9600} = 2621.44$s, soit 43.68 minutes.

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

Nom du processus	Durée (quantum)	Arrivée (quantum)	Priorité
P1	3	0	D (basse)
P2	2	1	D
P3	4	4	B
P4	2	5	B
P5	2	8	A (haute)

Table 1: Processus pour la question 2.

Indiquez quel processus sera exécuté à chaque quantum de temps pour chacun des algorithmes spécifiés ci-bas. Exécutez les étapes dans le même ordre que celui vu en classe, donc, pour chaque quanta:

1. Admission d'un nouveau processus (s'il y a lieu);
2. Choix du processus à exécuter selon l'algorithme d'ordonnancement;
3. (Tourniquet seulement): placer le processus à la fin de la file d'attente;
4. Passer au quanta suivant.

Si deux processus sont équivalents pour l'algorithme d'ordonnancement, commencez par celui *arrivé en premier*.

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

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

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

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

(c) (3 points) Tourniquet (avec file d'attente).

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

(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 (avec file d'attente). *Indice*: conservez une file d'attente différente pour chaque niveau de priorité.

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

3. (15 points) Votre voisin s'intéresse à l'informatique, mais n'a malheureusement jamais suivi le cours d'OSA. Fidèle à son habitude, il cogne chez vous à 7h le samedi matin et vous demande de l'aider à «débugger» son ordinateur. Quelle n'est pas votre surprise lorsque vous réalisez qu'il tente d'implémenter un système d'interruptions ARM grâce au simulateur du cours qu'il a trouvé sur le web! Dans cette question, vous aiderez votre voisin à trouver et à corriger ses erreurs.

   Avant d'aller chez lui, vous consultez tout d'abord vos notes de cours. Dans celles-ci, vous vous rappelez que la table des vecteurs d'interruptions en ARM est organisée ainsi:

Adresse	Interruption
`0x00`	Reset
`0x04`	Instruction indéfinie
`0x08`	Interruption logicielle
`0x0C`	«Prefetch abort»
`0x10`	«Data abort»
`0x14`	Espace réservé
`0x18`	IRQ
`0x1C`	FIQ

   Vous vous remémorez aussi que lors d'une interruption de type «FIQ», les registres `R8-R14` sont «banqués», c'est-à-dire qu'ils sont différents du programme principal en mode «User». Finalement, vous notez que les bits 31 à 28 du CPSR représentent les drapeaux «N», «Z», «C», et «V», respectivement.

   Armé(e) de ces informations, vous vous vêtez de vos plus beaux atours (pantoufles en phentex et robe de chambre), et vous vous dirigez à regret vers son ordinateur. Sur le chemin, il vous explique qu'il tente d'effectuer une boucle infinie, boucle qui sera interrompue par une interruption de type FIQ qui place le drapeau «Z» à zéro. Il vous mentionne que les interruptions FIQ sont configurées de façon appropriée dans le simulateur.

   Le code de votre voisin est disponible sur la page suivante. Après l'avoir analysé, vous parvenez à trouver cinq (5) erreurs.

   (a) (3 points) Vous décelez une première erreur en observant la table des vecteurs d'interruption. Quelle est cette erreur? Indiquez clairement la ou les lignes concernées. Proposez une solution pour la régler.

   > **Solution:** Le `B interruptionFIQ` n'est pas placé à la bonne adresse (0x14 au lieu de 0x1C) à la ligne 8. Il faudrait rajouter deux `NOP` entre le `B main` et `B interruptionFIQ` pour le mettre à la bonne adresse. (1 point pour l'erreur, 2 points pour la solution)

   (b) (3 points) Vous décelez une seconde erreur en observant la routine de traitement de l'interruption FIQ. Quelle est cette erreur? Indiquez clairement la ou les lignes concernées. Proposez une solution pour la régler.

   > **Solution:** Le CPSR est utilisé aux lignes 26 et 32, ce qui modifiera les drapeaux du mode «FIQ», et non du programme principal. Pour régler le problème, il suffit de remplacer CPSR par SPSR. (1 point pour l'erreur, 2 points pour la solution)

(c) (3 points) Vous décelez une troisième erreur en observant à nouveau la routine de traitement de l'interruption FIQ. Quelle est cette erreur? Indiquez clairement la ou les lignes concernées. Proposez une solution pour la régler.

Solution: La routine de traitement de l'interruption se termine par `BX LR` à la ligne 35, ce qui ne retournera pas à la bonne adresse, et ne replacera pas le micro-processeur dans le mode «User» du programme principal. Il faudrait plutôt utiliser `SUBS PC, LR, #4`. (1 point pour l'erreur, 2 points pour la solution)

(d) (3 points) Vous décelez une quatrième erreur en observant à nouveau la routine de traitement de l'interruption FIQ. Quelle est cette erreur? Indiquez clairement la ou les lignes concernées. Proposez une solution pour la régler.

Solution: Le bit «C» est mis à «0», et non «Z» à la ligne 29. Il faudrait remplacer par `AND R8, R8, #0xBFFFFFFF`. (1 point pour l'erreur, 2 points pour la solution)

(e) (3 points) Vous décelez une cinquième erreur en observant à nouveau la routine de traitement de l'interruption FIQ. Quelle est cette erreur? Indiquez clairement la ou les lignes concernées. Proposez une solution pour la régler.

Solution: Le registre `R0` est utilisé dans la routine de traitement de l'interruption (lignes 26, 29 et 32), mais il n'est pas banqué: cela pourrait donc affecter le programme principale. Pour régler le problème, nous pourrions utiliser une pile FIQ (assumant qu'une soit disponible), ou encore un des registres `R8-R12`. (1 point pour l'erreur, 2 points pour la solution)

Voici le code de votre voisin:

```
SECTION INTVEC

B main                ; interruption reset
NOP
NOP
NOP
NOP
B interruptionFIQ     ; interruption FIQ

SECTION CODE

main

MOV R0, #0
CMP R0, #0

boucleInfinie
BEQ boucleInfinie

; Suite du programme...

B main

interruptionFIQ
; accédons aux drapeaux du programme principal
MRS R0, CPSR

; placons le bit "Z" a 0 pour terminer la boucle
AND R0, R0, #0xDFFFFFFF

; appliquons nos changements au drapeaux du programme
MSR CPSR, R0

; fin de la routine de traitement de l'interruption FIQ
BX LR

SECTION DATA
```

Et dire qu'il ne vous a même pas offert le café!

4. (15 points) Dans un système avec mémoire paginée où:

   - les pages ont une taille de 32Ko;
   - la taille de la mémoire virtuelle est de 1Go;
   - la taille de la mémoire physique est de 128Mo;
   - un extrait de la table des pages est donné par :

Page	Trame
`0x00`	`0x01`
`0x01`	`0xB2`
`0x02`	`0xCD`
`0x03`	`0x05`
`0x04`	`0x9F`
`0x05`	`0x32`
`0x06`	`0x2D`
`0x07`	`0x7C`
`0x08`	`0x11`
`0x09`	`0x09`
`0x0A`	`0xBD`
⋮	⋮

   (a) (2 points) Quel est le nombre maximum de pages dans la table des pages pour ce système?

   Solution: $\frac{2^{30}}{2^{15}} = 2^{15} = 32768$ pages.

   (b) (2 points) Quel est le nombre de trames (*frames*) dans ce système?

   Solution: $\frac{2^{27}}{2^{15}} = 2^{12} = 4096$ trames.

   (c) (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: La taille totale de la table des pages sera donc de 32768×12 bits = 393216 bits = 49152 octets = 48 Ko. (1 point pour bonne réponse en bits, 1 point pour bonne réponse en octets, 4 points pour bonne réponse en Ko)

   (d) (4 points) Traduisez l'adresse virtuelle `0x29B35` en adresse physique en utilisant la table des pages ci-haut. Écrivez clairement votre démarche.

   Solution: L'adresse virtuelle `0x29B35` est séparée en deux: 1) les 15 LSB pour l'offset `0x1B35`, et 2) le reste pour le numéro de page `0x5` (1 point). Le numéro de trame correspondant à la page `0x5` est `0x32` (1 point), donc l'adresse résultante est `0x191B35` (2 points).

   (e) (2 points) À quelle *page* l'adresse *physique* `0x2A3C9` correspond-elle? Écrivez clairement votre démarche.

Solution: L'adresse physique `0x2A3C9` est séparée en deux: 1) les 15 LSB pour l'offset `0x23C9`, et 2) le reste pour le numéro de trame `0x5` (1 point). Le numéro de page correspondant à la trame `0x5` est `0x3` (1 point) selon la table des pages.

(f) (1 point) Dans un processeur ARM, quelle composante s'occupe de faire cette traduction?

Solution: C'est le MMU, ou le «Memory Management Unit».

5. (8 points) Un système informatique possède une mémoire totale de 64Ko et la gère avec une allocation contigüe avec partitions de taille variable. Les processus suivants sont alloués en mémoire:

   1. P1, 3Ko
   2. P2, 7Ko
   3. P3, 10Ko
   4. P4, 20Ko
   5. P5, 2Ko

   Après s'être exécutés pendant quelques temps, les processus P1, P2 et P4 se terminent.

   (a) (2 points) Quelles sont les adresses de début et de fin des processus P3 et P5 toujours en mémoire? Écrivez les adresses exactes en hexadécimal.

   Solution:
   P3: 0x2800 à 0x4FFF (10K à 20K-1)
   P5: 0xA000 à 0xA7FF (40K à 42K-1)
   (1 point pour bonnes adresses, 1 point pour réponse en hexa)

   Par la suite, les nouveaux processus suivants sont alloués, dans l'ordre:

   1. P6, 16Ko
   2. P7, 7Ko

   Quelles sont les adresses de début et de fin des processus P6 et P7 si chacun des algorithmes suivants sont utilisés pour allouer de l'espace aux nouveaux processus? Écrivez les adresses exactes en hexadécimal.

   (b) (2 points) meilleure allocation («best fit»)?

   Solution:
   P6: 0x5000 à 0x8FFF (20K à 36K-1)
   P7: 0x0000 à 0x1BFF (0K à 7K-1)
   (1 point pour bonnes adresses, 1 point pour réponse en hexa)

   (c) (2 points) prochaine allocation («next fit»)?

   Solution:
   P6: 0xA800 à 0xE7FF (42K à 58K-1)
   P7: 0x0000 à 0x1BFF (0K à 7K-1)
   (1 point pour bonnes adresses, 1 point pour réponse en hexa)

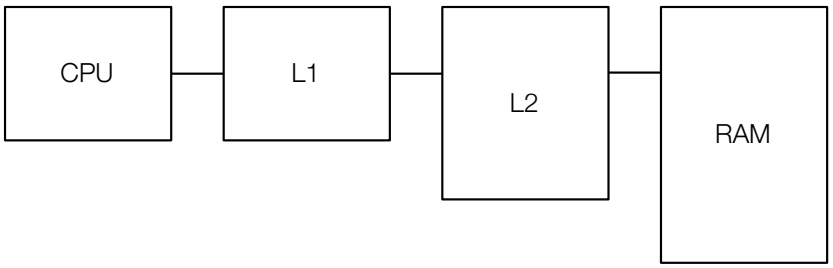
   (d) (2 points) pire allocation («worst fit»)?

   Solution:
   P6: 0xA800 à 0xE7FF (42K à 58K-1)
   P7: 0x5000 à 0x6BFF (20K à 27K-1)
   (1 point pour bonnes adresses, 1 point pour réponse en hexa)

6. (15 points) Un système possède les caractéristiques suivantes:

   - deux caches (L1 et L2) de type «write-back»;
   - les caches stockent des blocs contenant 8 mots;
   - la cache L1:
     - ne possède aucun bloc vide;
     - possède un bloc le moins récemment utilisé qui *n'est pas* sale;
   - la cache L2:
     - possède un seul bloc vide;
     - possède un bloc le moins récemment utilisé qui *est* sale;

   Ce système est illustré dans la figure suivante:



   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 est présent en cache L2, mais pas en cache L1.

   Solution: Le bloc n'est pas présent en cache L1, nous avons donc un «miss». Il nous faut donc:

   1. Trouver un bloc à remplacer en cache (par exemple grâce à l'algorithme LRU); ce bloc n'est pas sale alors il n'est pas écrit en L2;
   2. Copier le bloc requis de la cache L2 vers L1:
      (a) Le bloc est présent en cache L2, c'est un «hit»; on retourne ce bloc à la cache L1.
   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), après l'exécution des étapes 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».

(c) Le temps de transfert d'un bloc en RAM vers L2 prend 200ns, le temps de transfert d'un bloc en L2 vers L1 prend 75ns, et le temps de transfert d'un mot en L1 vers le CPU prend 5ns. Assumant que les blocs à remplacer ne sont jamais sales, calculez le temps total d'accès du CPU aux 8 mots d'un même bloc lorsque:

i. (1 point) 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: $8 \times 5\text{ns} = 40\text{ns}$.

ii. (1 point) Le bloc est en cache L2, mais pas en cache L1.

Solution: Le bloc doit être préalablement transféré en cache L1 à partir de la cache L2. Cela prend 75ns. Ensuite, on lit les 8 mots directement en L1: $75\text{ns} + 8 \times 5\text{ns} = 115\text{ns}$.

iii. (2 points) Le bloc est en RAM, mais n'est ni en cache L2 ni L1.

Solution: Le bloc doit être préalablement: transféré de la RAM vers L2 (200ns), et de L2 vers L1 (75ns). Ensuite, on lit les 8 mots directement en L1: $200\text{ns} + 75\text{ns} + 8 \times 5\text{ns} = 315\text{ns}$.

7. (12 points) Votre voisin (encore lui!) revient à la charge le dimanche matin (à 7h bien sûr) car il a encore des questions sur son code. Comme il promet de vous offrir le café pour se faire pardonner son oubli d'hier et comme vous avez un grand cœur, vous acceptez de l'aider.

   Cette fois, il tente d'implémenter un programme pour transférer des données du disque dur vers la mémoire. Son code fonctionne bien, mais il se demande comment améliorer ses performances. Vous débutez tout d'abord par analyser ce bout de code, qui effectue une requête au disque dur pour savoir quand il sera prêt à recevoir des données de la RAM:

```
1  ; Fait une requete au disque dur
2  LDR R0, =ControleDisqueDur
3  MOV R1, #1
4  STR R1, [R0]    ; Effectue la requete en envoyant un '1' au disque dur
5
6  LDR R0, =EtatDuDisqueDur
7  boucle
8  LDR R1, [R0]
9  TST R1, #1      ; si le bit 0 est a 1, le disque dur est pret
10 BNE boucle
```

   (a) (2 points) Décrivez, *sans écrire de code*, une alternative à la stratégie employée par votre voisin pour faire une requête au disque dur. Détaillez votre réponse.

   Solution: On peut remplacer la boucle d'interrogation («polling») par un système supportant les interruptions. Dans ce cas, le disque dur émet une interruption lorsqu'il est prêt. On doit rajouter une table des vecteurs d'interruption ainsi qu'une routine de traitement de l'interruption.

   (b) (2 points) Nommez un avantage de la solution que vous avez proposée en (a)?

   Solution: 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) (2 points) Nommez un inconvénient de la solution que vous avez proposée en (a)?

   Solution: L'architecture matérielle est plus compliquée: il nous faut un contrôleur d'interruptions, le micro-processeur doit supporter les interruptions (incluant changements de contexte, registres banqués, etc.)

Un peu plus loin, vous trouvez le bout de code suivant, qui gère le transfert mémoire entre la RAM et le disque dur une fois que le disque dur est prêt:

```
1  LDR R0, =AdresseSourceMemoire
2  LDR R3, =AdresseDestinationDisqueDur
3  LDR R1, NombreOctetsACopier
4
5  MOV R2, #0          ; Compteur
6  boucle
7  LDR R4, [R0], #4
8  STR R4, [R3], #4
9  ADD R2, R2, #1
10
11 CMP R2, R1
12 BNE boucle
13
14 ; et le programme continue...
```

(d) (2 points) Décrivez, *sans écrire de code*, une alternative à la stratégie employée par votre voisin pour effectuer un transfert mémoire entre le disque dur et la RAM. Détaillez votre réponse.

Solution: Le « 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.

(e) (2 points) Nommez un avantage de la solution que vous avez proposée en (d)?

Solution: Avantage: Le microprocesseur n'a pas à gérer le transfert des données, c'est le contrôleur de DMA qui le fait. Il peut donc faire autre chose pendant ce temps.

(f) (2 points) Nommez un inconvénient de la solution que vous avez proposée en (d)?

Solution: Inconvénients possibles: 1) architecture matérielle est plus compliquée car il faut rajouter le contrôleur de DMA; 2) les bus ne peuvent pas être partagés, le DMA doit donc attendre après le CPU lorsque ce dernier veut accéder aux bus (synchronisation plus compliquée).

8. (13 points) Répondez aux questions suivantes par une réponse courte.

   (a) (1 point) Quel est le nom de la cache spéciale du MMU pour la table des pages?

   Solution: Le «Translation Lookaside Buffer», soit le TLB.

   (b) (1 point) Où est situé le BIOS?

   Solution: Dans une ROM sur la carte mère.

   (c) (1 point) En général, quel bloc sera remplacé dans la cache lorsqu'elle est pleine et qu'un nouveau bloc doit être chargé?

   Solution: Le bloc le moins récemment utilisé («Least Recently Used»).

   (d) (1 point) En communication série, combien de mots de 8 bits peut-on envoyer à chaque seconde si la vitesse de transfert est de 19200 bits par seconde, et qu'on emploie la parité paire ainsi qu'un seul bit d'arrêt?

   Solution: $\frac{19200}{1+8+1+1} = 1745$

   (e) (1 point) À quoi servent les lignes D+ et D- dans le port USB?

   Solution: Pour transmettre le signal (D+) et l'inverse du signal (D-) en communication différentielle

   (f) (1 point) Lorsqu'un processus est dans l'état «bloqué», que doit-il survenir pour qu'il puisse retourner dans l'état «prêt»?

   Solution: La requête au périphérique doit se compléter.

   (g) (1 point) Le bus USB ne supporte pas les interruptions. De quelle façon sont-elles simulées?

   Solution: Par «polling», l'hôte interroge périodiquement tous les périphériques qui sont configurés pour générer des interruptions.

   (h) (1 point) Dans le protocole USB, pourquoi un paquet possède-t-il une adresse?

   Solution: Pour savoir à quel périphérique ce paquet est destiné.

   (i) (1 point) Pourquoi l'algorithme «premier arrivé, premier servi» n'est jamais utilisé dans un ordonnanceur moderne?

   Solution: Car il ne fait qu'exécuter les processus (au complet) dans l'ordre où ils arrivent: il ne répartit donc pas le temps du micro-processeur de façon à ce que les processus s'exécutent «en même temps».

(j) (1 point) Où est situé le premier stade du «bootloader» sur le disque dur?

Solution: Dans le premier secteur du disque, soit le «Master Boot Record» (MBR).

(k) (1 point) Comment est-ce que le micro-processeur sait quelle routine exécuter lorsqu'une interruption survient?

Solution: Il saute à une adresse de la table des vecteurs d'interruption, qui contient elle-même un branchement vers la bonne routine de traitement de l'interruption.

(l) (1 point) Quel est l'avantage d'une cache «write-back» par rapport à une cache «write-through»?

Solution: En «write-back», on écrit en RAM seulement lorsque c'est nécessaire (i.e. lorsqu'un bloc à remplacer est sale).

(m) (1 point) Pourquoi est-ce qu'un bus série peut être plus rapide qu'un bus parallèle, même s'il ne peut transférer qu'un seul bit à la fois?

Solution: La vitesse est limitée sur un bus en parallèle car il faut s'assurer que tous les bits soient reçus en même temps. La longueur des lignes doit être la même, ce qui est très difficile à atteindre. Comme un bus série n'a pas ces contraintes, sa vitesse peut être augmentée significativement!

A Annexe: Unités et logarithmes

A.1 Unités

Petit rappel sur les unités:

$$\begin{array}{lllll} 1\text{Ko} & = 2^{10} & & = & 1\ 024 \text{ octets} \\ 1\text{Mo} & = 2^{20} & = 1\ 024\text{Ko} & = & 1\ 048\ 576 \text{ octets} \\ 1\text{Go} & = 2^{30} & = 1\ 024\text{Mo} & = & 1\ 073\ 741\ 824 \text{ octets} \end{array}$$

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` / $2^{\texttt{imm}}$
`B etiquette`	`PC` ← `adresse(etiquette)`
`BL etiquette`	`LR` ← `PC - 4`, `PC` ← `adresse(etiquette)`
`BX Rs`	`PC` ← `Rs`
`CMP Rs, Op1`	Change les drapeaux comme `Rs - Op1`
`LDR Rd, =etiquette`	`Rd` ← `adresse(etiquette)`
`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, {C/S}PSR`	`Rd` ← `{C/S}PSR`
`MSR {C/S}PSR, Rs`	`{C/S}PSR` ← `Rs`
`MUL Rd, Rn, Rs`	`Rd` ← `Rn` × `Rs`
`MVN Rd, Op1`	`Rd` ← `!Op1` (inverse les bits)
`POP {Liste Reg}`	Charge les registres en ordre croissant à partir de la pile
`PUSH {Liste Reg}`	Met la liste de registres sur la pile dans l'ordre décroissant
`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.

C Annexe: Table ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	0	000	NUL	43	2B	053	+	86	56	126	V
1	1	001	SOH	44	2C	054	,	87	57	127	W
2	2	002	STX	45	2D	055	-	88	58	130	X
3	3	003	ETX	46	2E	056	.	89	59	131	Y
4	4	004	EOT	47	2F	057	/	90	5A	132	Z
5	5	005	ENQ	48	30	060	0	91	5B	133	[
6	6	006	ACK	49	31	061	1	92	5C	134	\
7	7	007	BEL	50	32	062	2	93	5D	135	]
8	8	010	BS	51	33	063	3	94	5E	136	^
9	9	011	TAB	52	34	064	4	95	5F	137	_
10	A	012	LF	53	35	065	5	96	60	140	‘
11	B	013	VT	54	36	066	6	97	61	141	a
12	C	014	FF	55	37	067	7	98	62	142	b
13	D	015	CR	56	38	070	8	99	63	143	c
14	E	016	SO	57	39	071	9	100	64	144	d
15	F	017	SI	58	3A	072	:	101	65	145	e
16	10	020	DLE	59	3B	073	;	102	66	146	f
17	11	021	DC1	60	3C	074	<	103	67	147	g
18	12	022	DC2	61	3D	075	=	104	68	150	h
19	13	023	DC3	62	3E	076	>	105	69	151	i
20	14	024	DC4	63	3F	077	?	106	6A	152	j
21	15	025	NAK	64	40	100	@	107	6B	153	k
22	16	026	SYN	65	41	101	A	108	6C	154	l
23	17	027	ETB	66	42	102	B	109	6D	155	m
24	18	030	CAN	67	43	103	C	110	6E	156	n
25	19	031	EM	68	44	104	D	111	6F	157	o
26	1A	032	SUB	69	45	105	E	112	70	160	p
27	1B	033	ESC	70	46	106	F	113	71	161	q
28	1C	034	FS	71	47	107	G	114	72	162	r
29	1D	035	GS	72	48	110	H	115	73	163	s
30	1E	036	RS	73	49	111	I	116	74	164	t
31	1F	037	US	74	4A	112	J	117	75	165	u
32	20	040	Space	75	4B	113	K	118	76	166	v
33	21	041	!	76	4C	114	L	119	77	167	w
34	22	042	"	77	4D	115	M	120	78	170	x
35	23	043	#	78	4E	116	N	121	79	171	y
36	24	044	$	79	4F	117	O	122	7A	172	z
37	25	045	%	80	50	120	P	123	7B	173	{
38	26	046	&	81	51	121	Q	124	7C	174	
39	27	047	’	82	52	122	R	125	7D	175	}
40	28	050	(	83	53	123	S	126	7E	176	~
41	29	051	)	84	54	124	T	127	7F	177	DEL
42	2A	052	*	85	55	125	U				