GIF-1001 Ordinateurs: Structure et Applications
Hiver 2018
Examen final
1er mai 2018
Durée : 180 minutes
Professeur : Jean-François Lalonde

Cet examen comporte 8 questions sur 17 pages (incluant celle-ci), comptabilisées sur un total de 100 points. L'examen compte pour 40% de la note totale pour la session.

- 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 ;
- Veuillez placer une carte d'identité sur votre table pour vérification ;
- Nous vous fournissons quatre (4) annexes :
  - l'annexe A (p. 14) contient des rappels sur la conversion d'unités et sur les logarithmes ;
  - l'annexe B (p. 15) contient une liste d'instructions ARM ainsi que des codes de conditions ;
  - l'annexe C (p. 16) contient la liste des registres banqués en assembleur ARM ;
  - l'annexe D (p. 17) contient la table ASCII.

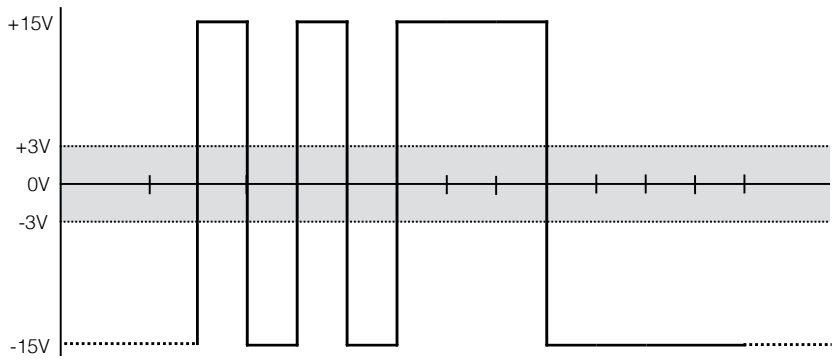
La table ci-dessous indique la distribution des points pour chaque question.

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

Bonne chance et bon été !

1. (15 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 ;
   — vitesse de transfert de 19 200 bits par seconde.

   (a) Le système envoie le signal illustré dans la figure ci-bas.



   i. (3 points) Sachant que la plage entre +3V et +15V correspond à un «0» logique, et que la plage entre -3V et -15V à un «1» logique, convertissez le signal en bits. Identifiez clairement : le bit de départ («start bit»), le bit de parité, ainsi que le bit d'arrêt («stop bit»).

> **Solution:** Les bits du mot sont envoyés du LSB au MSB. Les bits reçus sont : 0 (start bit) 1–0–1–0–0–0–1–1 (mot) 1 (parité) 1 (stop bit)

   ii. (2 points) Quel est le mot reçu en hexadécimal ?

> **Solution:** Le mot reçu est donc 0b11000101 soit 0xC5.

   (b) Le système envoie ensuite la série de bits suivante sur la ligne : 0–1–0–1–1–0–0–1–0–1–1. De l'autre côté, le périphérique détermine qu'il a reçu le mot 0x6E.

   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 0x4D et non 0x6E (1 point).

   ii. (3 points) Si le bit de parité reçu par le périphérique est «1», est-ce que le périphérique peut savoir s'il y a eu erreur de transmission ? Pourquoi ?

> **Solution:** Oui (1 point), car le nombre de «1» doit être impair, mais si le bit de parité est «1», alors c'est un nombre pair (2 points).

   (c) (3 points) On désire envoyer la chaîne de caractères «OSA» sur la ligne. Chaque caractère est encodé en ASCII (voir annexe D). Quelle sera la chaîne de bits à envoyer sur la ligne de transmission ?

Solution:
0–1–1–1–1–0–0–1–0–0–1 (1 point)
0–1–1–0–0–1–0–1–0–1–1 (1 point)
0–1–0–0–0–0–0–1–0–1–1 (1 point)

(d) (2 points) Le roman «Les Misérables» de Victor Hugo contient environ 513 000 mots. Si on suppose une moyenne conservatrice de 4 lettres par mot et que chaque mot est suivi d'un espace ou d'un signe de ponctuation, ce chef-d'oeuvre contiendrait environ 2 565 000 caractères.

On veut transmettre «Les Misérables» sur le lien série décrit ci-haut en encodant chaque caractère en ASCII (voir annexe D). Cependant, le lien est soumis à de fortes perturbations électromagnétiques, ce qui fait que 2% des caractères reçus sont erronés et doivent être retransmis. Vous pouvez assumer que les caractères erronés sont toujours détectés et simplement envoyés deux fois.

Combien de temps ce système prendra-t-il pour transférer «Les Misérables» sur un tel lien série ? Donnez votre réponse en minutes.

Solution: Chaque caractère est encodé sur 11 bits (1 point).
Temps total : $\frac{11 \times 2\ 565\ 000 \times 1.02}{19\ 200} = 1\ 498.92$s, soit 24,98 minutes (1 point).

2. (10 points) Les processus suivants sont admis, dans l'ordre.

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

Indiquez quel processus sera exécuté à chaque quantum de temps pour chacun des algorithmes spécifiés ci-bas.

Important : si deux processus sont équivalents (par exemple, deux processus ayant le même temps restant pour l'algorithme «plus court d'abord»), choisissez celui *arrivé en premier*.

(a) (1 point) Premier arrivé, premier servi (sans tenir compte de la priorité).

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

(b) (1 point) Plus prioritaire d'abord. Si deux (ou plus) processus ont la même priorité, utilisez le premier arrivé, premier servi.

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

(c) (2 points) Plus court d'abord (sans tenir compte de la priorité).

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

(d) (2 points) Plus prioritaire d'abord. Si deux (ou plus) processus ont la même priorité, utilisez le plus court d'abord.

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

(e) (2 points) Tourniquet (sans tenir compte de la priorité).

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

(f) (2 points) Plus prioritaire d'abord. Si deux (ou plus) processus ont la même priorité, utilisez le tourniquet. Dans ce cas, vous devez maintenir une file d'attente différente pour chaque niveau de priorité.

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

3. (15 points) Dans un système avec mémoire paginée où :
   — 17 bits sont nécessaires pour représenter la position d'une adresse dans une page ;
   — la taille de la mémoire virtuelle est de 2Go ;
   — la taille de la mémoire physique est de 256Mo ;
   — 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 de pages dans ce système ?

Solution: $\frac{2^{31}}{2^{17}} = 2^{14} = 16\ 384$ pages.

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

Solution: $\frac{2^{28}}{2^{17}} = 2^{11} = 2\ 048$ 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 $2^{14} \times 11$ bits = 180 224 bits = 22 528 octets = 22 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 `0x154236` en adresse physique en utilisant la table des pages ci-haut. Écrivez clairement votre démarche.

Solution: L'adresse virtuelle `0x154236` est séparée en deux : 1) les 17 LSB pour l'offset `0x14236`, et 2) le reste pour le numéro de page `0xA` (1 point). Le numéro de trame correspondant à la page `0xA` est `0xBD` (1 point), donc l'adresse résultante est `0x17B4236` (2 points).

(e) (2 points) À quelle *page* l'adresse *physique* `0x5A3CF9` correspond-elle ? Écrivez clairement votre démarche.

Solution: L'adresse physique `0x5A3CF9` est séparée en deux : 1) les 17 LSB pour l'offset `0x3CF9`, et 2) le reste pour le numéro de trame `0x2D` (1 point). Le numéro de page correspondant à la trame `0x2D` est `0x6` (1 point) selon la table des pages.

(f) (1 point) Combien y a-t-il d'éléments dans la table des pages *inversée* de ce système ?

Solution: Comme il y a 2 048 trames, il y aurait 2 048 lignes dans la table des pages inversée.

4. (10 points) La liste suivante indique des processus en attente d'allocation mémoire (dans l'ordre de leur arrivée), ainsi que leur taille. La taille totale de la mémoire physique du système est de 64Ko.

   1. P1, 7Ko
   2. P2, 2Ko
   3. P3, 10Ko
   4. P4, 8Ko
   5. P5, 5Ko

   Commençons tout d'abord par analyser le cas d'une architecture utilisant la stratégie d'allocation par mémoire contigüe avec partitions de taille *fixe*. Les partitions ont une taille de 8Ko.

   (a) (2 points) Est-ce que le système parvient à allouer de l'espace à tous les processus ? Sinon, quel processus ne peut être chargé ?

   Solution: Non, P3 est trop gros et ne peut être alloué.

   (b) (2 points) Quelle sera la fragmentation interne totale lorsque tous les processus pouvant être chargés seront en mémoire ?

   Solution: La fragmentation interne totale sera de $(8-7)+(8-2)+(8-8)+(8-5) = 1+6+3 = 10$Ko.

Analysons maintenant le cas d'un autre système, toujours à mémoire contigüe, mais qui utilise cette fois des partitions de taille *variable*. Les mêmes processus ci-haut sont tout d'abord admis en mémoire. Ensuite, les processus P1 et P3 se terminent. Finalement, les nouveaux processus suivants surviennent, dans l'ordre :

1. P6, 8Ko
2. P7, 2Ko

(c) (3 points) Écrivez le contenu de la mémoire si l'algorithme suivant est utilisé pour allouer de l'espace aux nouveaux processus : meilleure allocation («best fit») ?

Solution:

Adresses	Processus
7–9	P2
9–17	P6
17–19	P7
19–27	P4
27–32	P5

(d) (3 points) Écrivez le contenu de la mémoire si l'algorithme suivant est utilisé pour allouer de l'espace aux nouveaux processus : prochaine allocation («next fit») ?

Solution:

Adresses	Processus
7–9	P2
19–27	P4
27–32	P5
32–40	P6
40–42	P7

5. (10 points) En 2019, les responsables du cours GIF-1001 ont bonifié le simulateur ARM pour simuler les communications avec les périphériques. Comme vous avez bien réussi le cours en 2018, vous êtes engagé(e) comme dépanneur(e). Votre première tâche est de concevoir un nouveau TP qui permet de simuler les interactions avec un scanneur, soit la numérisation d'une page, et le transfert de cette page numérisée vers la RAM.

   Vous écrivez le code suivant pour remplir une partie de la tâche :

```
1  LDR R0, =AdresseScanneur
2  LDR R3, =AdresseEnRAM
3  LDR R1, NombreOctetsACopier
4
5  PUSH {R5}
6  MOV R2, #0
7  boucle
8  LDR R5, [R0], #4
9  STR R5, [R3], #4
10 ADD R2, R2, #1
11
12 CMP R2, R1
13 BNE boucle
14 POP {R5}
```

(a) (2 points) Décrivez, *sans écrire de code*, une alternative à la stratégie employée ci-haut pour communiquer avec le scanneur. 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.

(b) (3 points) Nommez un avantage et un inconvénient de la solution que vous avez proposée en (a) par rapport à la stratégie employée par le code ci-haut ?

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.

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).

Vous écrivez cet autre code pour remplir une autre partie de la tâche :

```
1  ; Demande au scanneur de numeriser une page.
2  ; Pour ce faire, on lui envoie le code "0xAB"
3  LDR R0, =ControleScanneur
4  MOV R3, #0xAB
5  STR R3, [R0]
6
7  LDR R0, =EtatScanneur
8  boucle
9  LDR R3, [R0]
10 CMP R3, #1      ; si le bit 0 est a 1, le scanneur a termine
11 BNE boucle
```

(c) (2 points) Décrivez, *sans écrire de code*, une alternative à la stratégie employée ci-haut pour communiquer avec le scanneur. Détaillez votre réponse.

Solution: On peut remplacer la boucle d'interrogation («polling») par un système supportant les interruptions. Dans ce cas, le scanneur 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.

(d) (3 points) Nommez un avantage et un inconvénient de la solution que vous avez proposée en (c) par rapport à la stratégie employée par le code ci-haut ?

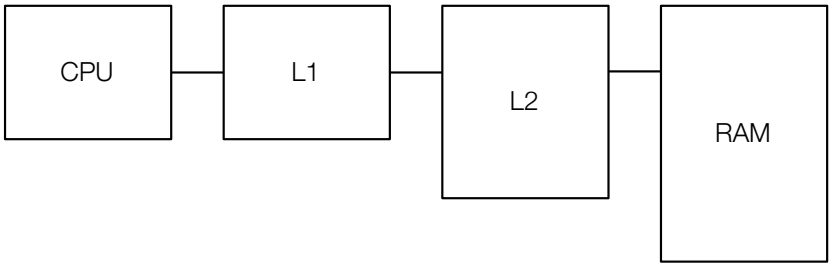
Solution: Avantage : le CPU peut faire autre chose en attendant que le périphérique soit prêt, ce qui n'est pas le cas des entrées-sorties programmées. C'est le périphérique qui indique quand il est prêt via une interruption.

Inconvénients possibles : 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.)

6. (15 points) Un système possède les caractéristiques suivantes :
   — deux caches (L1 et L2) ;
   — les blocs en cache contiennent 8 mots ;
   — la cache L1 :
     — est de type «write-back» ;
     — ne possède aucun bloc vide ;
     — possède un bloc le moins récemment utilisé qui *n'est pas* sale ;
   — la cache L2 :
     — est de type «write-through» ;
     — ne possède aucun bloc vide ;

   Ce système est illustré dans la figure suivante :



   Répondez aux questions suivantes portant sur ce système.

   (a) (3 points) Décrivez les étapes nécessaires à l'*écriture* d'un mot d'un bloc qui est présent en cache L1.

   Solution: Le bloc est présent en cache L1, nous avons donc un «hit». Il faut :
   1. Écrire le mot dans le bloc en question ;
   2. Marquer le bloc comme étant «dirty».

   (b) (4 points) Décrivez les étapes nécessaires à l'*écriture* d'un 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 faut :
   1. Trouver un bloc à remplacer grâce à LRU (il n'est pas sale) ;
   2. Copier le bloc de la cache L2 dans la cache L1 :
      (a) Le bloc est présent en cache L2, c'est un «hit» ;
      (b) On peut lire les données et les copier vers L1.
   3. Écrire le mot en cache L1 ;
   4. Marquer le bloc comme étant sale.

   (c) (2 points) Si la cache L2 était «write-back» plutôt que «write-through», qu'aurait-il fallu faire différemment pour l'écriture d'un mot d'un bloc présent en cache L2 mais pas en L1, comme en (b) ?

Solution: Comme le bloc est présent en L2 et que la cache L1 est «write-back», les étapes seraient exactement les mêmes !

(d) Sachant que le temps de transfert
— d'un *mot* de la cache L1 vers le CPU est de 4ns ;
— d'un *bloc* de la cache L2 vers la cache L1 est de 64ns ;
— d'un *bloc* de la RAM vers la cache L2 est de 256ns ;

i. (2 points) Quel temps sera nécessaire pour accéder aux 8 mots d'un même bloc qui n'est pas en cache ?

Solution: Le bloc doit être préalablement : transféré de la RAM vers la cache L2 (256ns), et de la cache L2 vers la cache L1 (64ns) (1 point). Ensuite, on lit les 8 mots directement en cache L1. Donc : $256 + 64 + 8 \times 4 = 352$ ns (1 point).

ii. (2 points) Quel temps sera nécessaire pour accéder à 8 mots répartis sur 4 blocs qui sont en cache L2 mais pas en cache L1 ?

Solution: Les blocs doivent être préalablement transférés en cache L1 à partir de la cache L2 ($64 \times 4 = 256$ ns) (1 point). Ensuite, on lit les 8 mots directement en L1 ($8 \times 4 = 32$ ns) (1 point). Donc, au total, 288ns.

iii. (2 points) Quel temps sera nécessaire pour accéder à 4 mots répartis sur 4 blocs qui sont tous en cache L1 ?

Solution: Il suffit de lire les 4 mots directement : $4 \times 4 = 16$ ns.

7. (15 points) Toujours dans le cadre de votre mandat de dépanneur(e) du cours GIF-1001 à l'hiver 2019 (voir question 5), vous vous voyez confier la tâche d'assister les étudiants durant une période de dépannage au PLT-0103. Pour vous préparer à votre période de dépannage, vous révisez les notes de cours qui vous rappellent que :

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

Lors d'un TP sur les interruptions, un étudiant vient vous voir car son code ne fonctionne pas. Le code de l'étudiant est disponible sur la page suivante. Après l'avoir analysé, vous parvenez à trouver cinq (5) erreurs.

(a) (3 points) Vous décelez une erreur dans la table des vecteurs d'interruption. Décrivez l'erreur, identifiez clairement la ou les lignes concernées, et proposez une solution pour la régler.

Solution: Le `B interruptionFIQ` n'est pas placé à la bonne adresse (0x18 au lieu de 0x1C) à la ligne 8. Il faudrait rajouter un `NOP` entre le `B interruptionSVC` 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 erreur dans le programme principal (lignes 13–24). Décrivez l'erreur, identifiez clairement la ou les lignes concernées, et proposez une solution pour la régler.

Solution: La boucle d'attente est faite avec un branchement *inconditionnel* avec l'instruction `B` à la ligne 20. La boucle ne se terminera donc jamais. Il aurait plutôt fallu faire `BEQ boucleAttente`. (1 point pour l'erreur, 2 points pour la solution)

(c) (3 points) Vous décelez une erreur dans la routine de traitement de l'interruption SVC. Décrivez l'erreur, identifiez clairement la ou les lignes concernées, et proposez une solution pour la régler.

Solution: Le registre `R14` est utilisé pour envoyer la requête au disque dur aux lignes 29 et 30, mais ce registre est LR ! S'il est modifié alors le `SUBS PC, LR, #4` ne fonctionnera pas. Il faudrait employer un autre registre (par exemple `R1`) sans oublier de le sauvegarder et le restaurer par la suite ! (1 point pour l'erreur, 2 points pour la solution)

(d) (3 points) Vous décelez une autre erreur dans la routine de traitement de l'interruption SVC. Décrivez l'erreur, identifiez clairement la ou les lignes concernées, et proposez une solution pour la régler.

Solution: Pour terminer le traitement de l'interruption, le `SUBS PC, LR, -#4` soustrait -4 (donc *additionne* 4) à `PC`. Il faudrait retirer le négatif : `SUBS PC, LR, #4`, ou encore utiliser `ADDS PC, LR, -#4`. (1 point pour l'erreur, 2 points pour la solution)

(e) (3 points) Vous décelez *deux* erreurs dans la routine de traitement de l'interruption FIQ. Décrivez *une de ces deux erreurs* (celle de votre choix), identifiez clairement la ou les lignes concernées, et proposez une solution pour la régler.

Solution: Première erreur : l'appel de la fonction `copieMemoire` est fait avec l'instruction `B` (ligne 41). Il faudrait plutôt utiliser `BL`. (1 point pour l'erreur, 2 points pour la solution)

Deuxième erreur : la routine de traitement de l'interruption se termine par `BX LR` à la ligne 44, 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)

Voici le code de l'étudiant :

```
SECTION INTVEC

B main                ; interruption reset
NOP
B interruptionSVC     ; interruption logicielle
NOP
MOV R0, #0
NOP
B interruptionFIQ     ; interruption FIQ

SECTION CODE

main            ; Debut du code principal

MOV R0, #0
SVC #10         ; Appel systeme pour demarrer une requete au disque dur

boucleAttente
CMP R0, #0
B boucleAttente

; Le code se continue...
fin
B fin

interruptionSVC

LDR R13, =ControleDisqueDur
MOV R14, #0x1
STR R14, [R13]   ; envoyer la requete au disque dur

SUBS PC, LR, #-4 ; fin de la routine de traitement de l'interruption SVC

interruptionFIQ

; charger les bonnes adresses
LDR R8, =AdresseDisqueDur
LDR R9, =AdresseEnRAM
LDR R10, NombreOctetsACopier

B copieMemoire  ; copier la memoire
MOV R0, #1      ; terminer la boucle

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

copieMemoire
; cette fonction est ecrite sans erreur ici...
BX LR
```

8. (10 points) Répondez aux questions suivantes par une réponse courte.
   (a) (1 point) Lorsqu'il y a une erreur de transmission en communication série, comment peut-on faire pour savoir quel bit est fautif ?

Solution: On ne peut pas le savoir. Le bit de parité ne sert qu'à indiquer si une erreur a été commise, il ne peut pas être utilisé pour déceler l'erreur.

(b) (1 point) Quel est le rôle principal du MMU ?

Solution: La traduction d'adresses, soit le calcul de l'adresse physique en RAM correspondant à l'adresse virtuelle du programme.

(c) (1 point) En mémoire paginée, quelle information est stockée dans la table des pages *inversée* ?

Solution: Le numéro de la page stockée dans chaque trame.

(d) (1 point) Pourquoi y a-t-il deux lignes dédiées au transport des données dans le bus USB ?

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

(e) (1 point) Vrai ou faux ? Après être dans l'état «bloqué», un processus devient dans l'état «en cours» dès que la requête du périphérique est terminée.

Solution: Faux. Il reprend l'état «prêt».

(f) (1 point) Pourquoi l'ordre d'exécution de deux boucles imbriquées est-il important ?

Solution: Car on doit parcourir les éléments dans le même ordre qu'ils sont stockés en mémoire pour augmenter le nombre de «hits» en cache.

(g) (1 point) À quoi sert un contrôleur de périphérique ?

Solution: À faire le «pont» entre le périphérique et les bus du microprocesseur.

(h) (1 point) Vrai ou faux ? La mémoire virtuelle doit avoir la même taille que la mémoire physique.

Solution: Faux. Elle est généralement plus grande !

(i) (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.

(j) (1 point) Nommez un inconvénient d'une cache «write-back» par rapport à une cache «write-through» ?

Solution: Il faut identifier les blocs sales, et les blocs doivent être écrits seulement s'ils sont sales. Ceci est plus compliqué qu'en «write-through» et requiert de la logique additionnelle.

A Annexe : Unités et logarithmes

A.1 Unités

Petit rappel sur les unités :

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

A.2 Logarithme en base 2

Pour calculer des logarithmes en base 2 à partir de logarithmes dans une autre base N (ex : 10), 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, #cte`	`Rd` ← `Rs` / $2^{\texttt{cte}}$
`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` ← `valeur(etiquette)`
`LDR Rd, =etiquette`	`Rd` ← `adresse(etiquette)`
`LDR Rd, [Rb, Op1]`	`Rd` ← `Mem[Rb + Op1]`
`LDR Rd, [Rb], Op1`	`Rd` ← `Mem[Rb]`, `Rb` ← `Rb + Op1`
`LDR Rd, [Rb, Op1]!`	`Rb` ← `Rb + Op1`, `Rd` ← `Mem[Rb]`
`LSL Rd, Rb, #cte`	`Rd` ← `Rb` × $2^{\texttt{cte}}$
`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, `SP` ← `SP - 4` × `(nombre de registres)`
`PUSH {Liste Reg}`	`SP` ← `SP + 4` × `(nombre de registres)`, Met la liste de registres sur la pile dans l'ordre décroissant
`STR Rs, etiquette`	`valeur(etiquette)` ← `Rd`
`STR Rs, [Rb, Op1]`	`Mem[Rb + Op1]` ← `Rs`
`STR Rs, [Rb], Op1`	`Mem[Rb]` ← `Rs`, `Rb` ← `Rb + Op1`
`STR Rs, [Rb, Op1]!`	`Rb` ← `Rb + Op1`, `Mem[Rb]` ← `Rs`
`SUB Rd, Rs, Op1`	`Rd` ← `Rs - Op1`

TABLE 1 – Instructions ARM. `Op1` dénote une opérande de type 1, soit une constante, un registre ou un registre décalé.

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 2 – Codes de condition.

C Annexe : Registres banqués en assembleur ARM

Dans la figure ci-bas, les registres possédant un triangle grisé sont dits «banqués» dans un microprocesseur ARM.

System and User	FIQ	Supervisor	Abort	IRQ	Undefined
r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12
r13 (SP)	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und
r14 (LR)	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und
r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

D 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				