GIF-1001 Ordinateurs: Structure et Applications
Hiver 2016
Examen mi-session
23 février 2016
Durée: 110 minutes

Cet examen comporte 7 questions sur 9 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;
- L'annexe A contient une liste 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:	15	15	15	15	20	10	10	100

Bonne chance!

1. Répondez aux questions suivantes sur la façon dont les données sont stockées dans un ordinateur.
   (a) (2 points) Effectuez l'addition suivante sur 8 bits en complément-2: $-5 + 5$, et indiquez s'il y a débordement.
   (b) (3 points) En complément-2, comment peut-on faire pour détecter un débordement? Donnez un exemple sur 4 bits.
   (c) (5 points) Sachant que 0x41 représente le caractère "A" en ASCII et le nombre 65 en décimal, comment peut-on savoir laquelle de ces deux valeurs est la bonne en l'absence de toute autre information?
   (d) (2 points) Quel est l'équivalent en décimal de la chaîne binaire 0x42280000 encodée sur 32 bits avec IEEE 754? Rappel: la norme IEEE 754 emploie la formule

$$(\text{signe})1, \text{mantisse} \times 2^{(\text{exposant}-127)},$$

   et les bits sont stockés selon la figure 1.

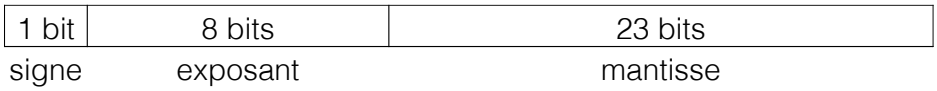

Figure 1: Convention IEEE-754 sur 32 bits.

   (e) (3 points) Lequel de ces deux nombres, tous deux encodés sur 32 bits avec la norme IEEE 754, est le plus grand des deux: 0x8A0153FA et 0x1A0153FA? Indiquez pourquoi.

2. Le schéma de la figure 2 représente l'architecture interne d'un microprocesseur simple, comme nous l'avons vu dans le cours. Utilisez le pour répondre aux questions suivantes.

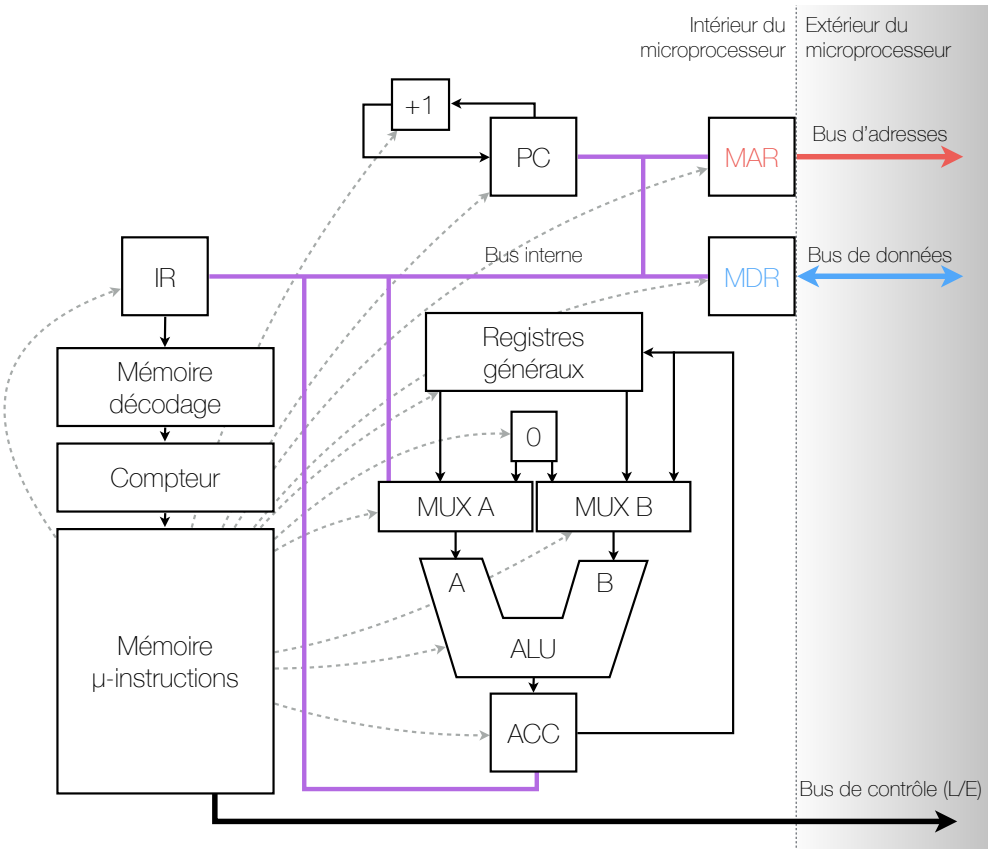

Figure 2: Architecture interne d'un microprocesseur simple pour la question 2.

(a) (1 point) À quoi sert le registre `IR`?

(b) (2 points) À quoi servent les registres `MAR` et `MDR`?

(c) (2 points) Quelle est l'utilité des "MUX" connectés sur l'ALU?

(d) (3 points) Décrivez les trois signaux de contrôle qui doivent être activés par la mémoire de micro-instruction lorsque la micro-instruction suivante est exécutée: `ACC` $\leftarrow$ `R0` $+ 0$?

(e) (7 points) Écrivez les micro-instructions qui correspondent à la lecture et à l'exécution de l'instruction `ADD R1 #0x02` (`R1` = `R1` + 0x02). Vous pouvez utiliser la notation "`IR` $\leftarrow$ `PC`" pour représenter un déplacement du contenu du registre `PC` vers le registre `IR`, par exemple. N'oubliez pas d'indiquer les signaux du bus de contrôle, s'il y a lieu.

3. Répondez aux questions suivantes, portant sur l'assembleur ARM.

   (a) (5 points) Considérez le code suivant (les numéros de ligne sont indiqués à gauche):

```
1    MOV LR, PC
2  test
3    BX LR
4    BL test
5    CMP LR, PC
6    BLT test
7    MOV R0, #0x1
8    B test
```

   Indiquez l'ordre des instructions exécutées par le microprocesseur en utilisant leur numéro de ligne correspondant.

   (b) (5 points) Écrivez du code assembleur qui place dans `R0` le signe (+ ou -1) de `R1`. En d'autres mots, implémentez le pseudo-code suivant:

if *R1* < 0 **then**
  `R0` $\leftarrow -1$;
else
  `R0` $\leftarrow 1$;
end

   (c) (5 points) Écrivez du code assembleur qui calcule la factorielle d'un nombre placé dans `R0`. En d'autres mots, implémentez le pseudo-code suivant:

`R1` $\leftarrow 1$;
while *R0* $\neq 0$ **do**
  `R1` $\leftarrow$ `R1` $\times$ `R0` ;
  `R0` $\leftarrow$ `R0` $- 1$;
end

4. Répondez aux questions portant sur le code assembleur ARM suivant (les numéros de ligne sont indiqués à gauche):

```
1   B       main
2
3  tableau DC32 0xFA, 0x32, 0x05, 0x45, 0x02, 0x00
4
5  main
6    LDR SP, =maPile
7    ADD SP, SP, #64
8
9    LDR R0, =tableau
10   BL fonctionMystere
11   MOV R5, R0
12
13   B main
14
15 fonctionMystere
16   PUSH {R1, R2, LR}
17   LDR R1, [R0], #4
18   MOV R2, R1
19
20 debut
21   CMP R1, #0x00
22   BEQ fin
23
24   CMP R1, R2
25   MOVLT R2, R1
26   LDR R1, [R0], #4
27   B debut
28
29 fin
30   MOV R0, R2
31   POP {R1, R2, LR}
32   BX LR
33
34 maPile DS32  16
```

(a) (1 point) Pourquoi utilise-t-on l'instruction `BL` et non `B` à la ligne 10?

(b) (2 points) Pourquoi utilise-t-on les instructions `PUSH` et `POP` aux lignes 16 et 31?

(c) (3 points) Que fait l'instruction `LDR R1, [R0], #4` à la ligne 17?

(d) (4 points) Quelle est la valeur de `R5` après l'exécution de l'instruction `MOV R5, R0` à la ligne 11?

(e) (5 points) Décrivez succinctement ce que fait la fonction `fonctionMystere`. De plus, indiquez ses arguments, la façon dont elle les obtient du programme principal, et comment elle retourne sa valeur de sortie.

5. Un système de type “memory-mapped I/O” possède les caractéristiques suivantes:

   - un bus d'adresse de 16 bits, avec les 2 bits les plus significatifs (MSB) utilisés pour le décodeur d'adresse;
   - un bus de données de 16 bits;
   - une mémoire RAM où chaque octet possède une adresse différente;
   - trois autres périphériques sont branchés sur les bus;
   - il stocke les données en mémoire avec la convention “big endian”;
   - si on nomme les bits les plus significatifs (MSB) du bus d'adresse b_{14} et b_{15}, le décodeur sélectionne les périphériques de la façon suivante:

b_{15}	b_{14}	Périphérique activé
0	0	RAM
0	1	Périphérique 1
1	0	Périphérique 2
1	1	Périphérique 3

   (a) (2 points) Quelle est la taille maximale de la mémoire RAM? Écrivez votre réponse en kilo-octets (Ko).

   (b) (3 points) On emploie une instruction pour stocker la valeur 0x1234 à l'adresse 0x0 de la mémoire RAM. Indiquez la ou les adresse(s) mémoire affectée(s) par cette instruction, ainsi que le contenu de la RAM après l'exécution de cette instruction.

Nous modifions le système afin qu'il puisse utiliser un bus de données de 32 bits au lieu de seulement 16. Le bus d'adresse ne change pas.

   (c) (2 points) Décrivez l'impact de ce changement sur la taille maximale de la mémoire RAM. Écrivez votre réponse en kilo-octets (Ko).

   (d) (3 points) Décrivez l'impact de ce changement sur l'écriture de la valeur 0x1234 à l'adresse 0x0 de la mémoire RAM. Comme précédemment, indiquez la ou les adresse(s) mémoire affectée(s) par cette instruction, ainsi que le contenu de la RAM après l'exécution de cette instruction.

   (e) (5 points) Quelle est la carte mémoire de ce système?

Finalement, modifions le système à nouveau pour qu'il puisse utiliser un bus d'adresse de 32 bits au lieu de seulement 16. Le système possède maintenant un bus de données de 32 bits, et un bus d'adresse de 32 bits. Les deux MSB b_{30} et b_{31} sont maintenant utilisés pour le décodeur d'adresse.

   (f) (5 points) Quelle est la carte mémoire de ce système?

6. Répondez aux questions suivantes portant sur le micro-processeur du simulateur du travail pratique 1. Le jeu d'instructions de ce micro-processeur est illustré en figure 3.

Toutes les instructions du microprocesseur sont sur 16 bits et se décomposent comme suit :
Bits 15 à 12 : Opcode de l'instruction
Bits 11 à 8 : Registre utilisé comme premier paramètre.
Bits 7 à 0 : Registre ou constante utilisés comme deuxième paramètre
Le microprocesseur possède quatre registres généraux nommés R0, R1, R2 et R3. En plus de ces quatre registres, un registre de pointeur d'instruction PC est disponible. Cependant, ce registre ne peut être utilisé qu'avec l'instruction MOV. Le nombre identifiant le registre PC est 0xF (15).

Le jeu d'instruction supporte les instructions suivantes où Rd est le registre destination, Rs le registre source et Rc le registre de condition :

Mnémonique	**Opcode**	**Description**
MOV Rd Rs	0000	Écriture de la valeur du registre Rs dans le registre Rd
MOV Rd Const	0100	Écriture d'une constante dans le registre Rd
ADD Rd Rs	0001	Addition des valeurs des registres Rd et Rs et insertion du résultat dans le registre Rd
ADD Rd Const	0101	Addition de la valeur du registre Rd avec une constante et insertion du résultat dans Rd
SUB Rd Rs	0010	Soustraction de la valeur Rs à l'intérieur de registre Rd.
SUB Rd Const	0110	Soustraction d'une constante à l'intérieur du registre Rd
LDR Rd [Rs]	1000	Chargement d'une valeur se trouvant à l'adresse Rs de l'ordinateur dans un registre.
STR Rd [Rs]	1001	Écriture de la valeur d'un registre à l'adresse Rs de l'ordinateur.
JZE Rc Const	1111	Saut à l'instruction située à l'adresse identifiée par la constante, mais seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).
JZE Rc [Rs]	1011	Saut à l'instruction située à l'adresse Rs seulement si Rc = 0 (sinon, cette instruction n'a aucun effet).

Figure 3: Jeu d'instructions du micro-processeur pour la question 6.

(a) (5 points) Traduisez le programme suivant en binaire, et écrivez votre réponse en hexadécimal. Les numéros de ligne sont indiqués à gauche.

```
1 MOV R0 #0x70
2 LDR R1 [R0]
3 ADD R0 #1
4 LDR R1 [R0]
5 JZE R1 [R2]
```

(b) (5 points) Que fait le programme suivant? Pour chaque ligne, on indique l'adresse (qui commence à `0x0`), suivie de l'instruction en format binaire. Les numéros de ligne sont indiqués à gauche.

```
1 0x0   0x4080
2 0x1   0x8100
3 0x2   0xF105
4 0x3   0x6101
5 0x4   0x4F02
6 0x5   0x9100
```

7. Répondez aux questions suivantes par vrai ou faux.
   (a) (1 point) Un pipeline à trois étages permet à un micro-processeur de lire, décoder, et exécuter la même instruction de façon simultanée.
   (b) (1 point) Dans un ordinateur RISC, toutes les instructions ont la même taille.
   (c) (1 point) Une pile est une structure de données de type FIFO, “first in, first out”.
   (d) (1 point) Dans une architecture de type “memory-mapped I/O”, le bus de contrôle sert à déterminer quel périphérique est activé.
   (e) (1 point) Lors de l’exécution d’une instruction `LDR`, le bus de contrôle est placé en lecture.
   (f) (1 point) Un des avantage des architectures CISC est que leur consommation d’énergie est réduite par rapport à RISC.
   (g) (1 point) Le langage assembleur facilite l’écriture d’un programme pour un microprocesseur donné.
   (h) (1 point) L’instruction ARM `LDR R0, =maVariable` place le contenu de `maVariable` dans `R0`.
   (i) (1 point) Dans l’instruction ARM `MOV R0, #0x70`, le `#` indique que `0x70` est une adresse.
   (j) (1 point) L’hexadécimal est une façon plus compacte de représenter du binaire.

A 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` × $2^{\texttt{imm}}$
`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 1: 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 2: Codes de condition.