

GIF-1001 Ordinateurs: Structure et Applications
Hiver 2015
Examen mi-session
24 février 2015
Durée: 110 minutes

Cet examen comporte 10 questions sur 7 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	8	9	10	Total
Points:	15	10	10	10	5	10	10	10	10	10	100

La question 9 contient aussi 5 points bonus.

Bonne chance!

1. Registres spéciaux en assembleur ARM: SP, PC et RL.
 - (a) (5 points) Qu'est-ce qu'indique le pointeur de pile (SP)? Quelles sont les deux instructions qui l'utilisent? Comment sa valeur est-elle modifiée lorsqu'on empile une donnée sur la pile?
 - (b) (5 points) Quelle est la taille du registre PC en ARM? Que contient-il? De combien doit-il être incrémenté à chaque instruction? Peut-on y accéder avec une instruction MOV?
 - (c) (5 points) À quoi sert le registre de liens (RL)? Donnez un exemple de son utilisation.
2. Répondez aux questions suivantes sur l'ALU.
 - (a) (2 points) Qu'est-ce qu'un ALU, et où est-il situé?
 - (b) (3 points) Décrivez trois drapeaux de l'ALU.
 - (c) (5 points) Expliquez, à l'aide d'un exemple en assembleur ARM, à quoi servent les drapeaux de l'ALU dans un microprocesseur.
3. Le schéma de la figure 1 représente l'architecture interne d'un microprocesseur simple, comme nous l'avons vu dans le cours. Utilisez-le pour répondre aux questions suivantes.

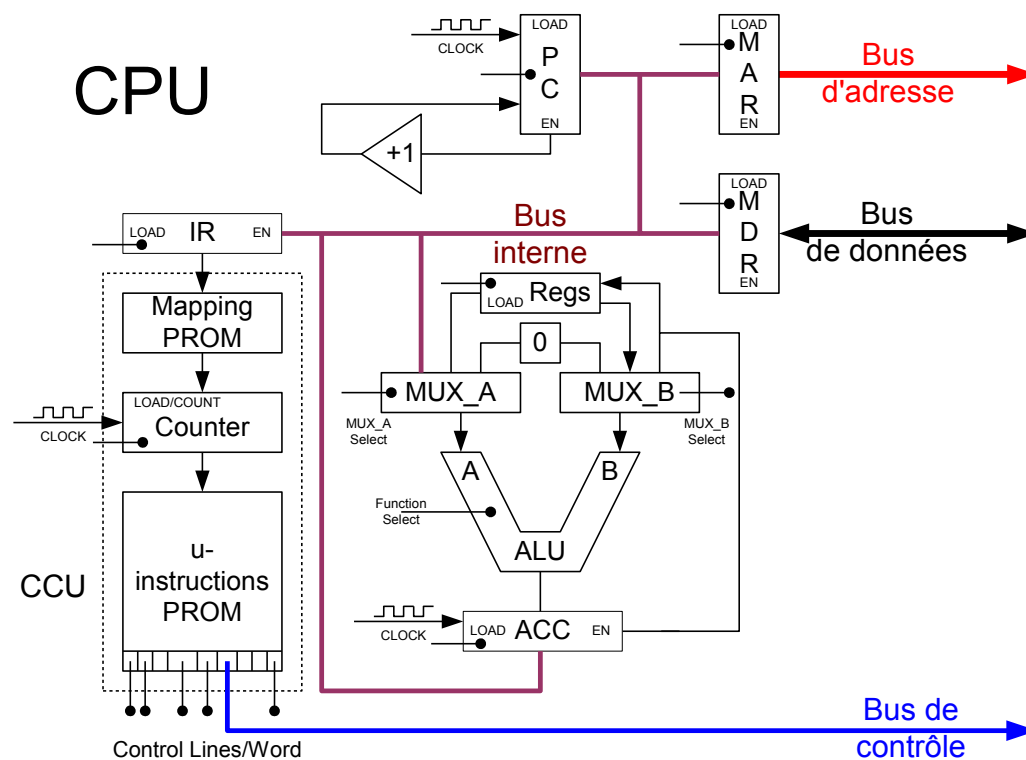


Figure 1: Architecture interne d'un microprocesseur simple pour la question 3.

- (a) (3 points) Nommez et décrivez brièvement les trois étapes d'exécution d'une instruction dans un microprocesseur.
- (b) (7 points) En utilisant le schéma de la figure 1, donnez les micro-instructions qui correspondent à l'instruction `STR R2 [R3]`. Vous pouvez utiliser la notation "`PC → IR`" pour représenter un déplacement du contenu du registre PC vers le registre IR, par exemple.

4. Répondez aux questions suivantes sur la façon dont les données sont stockées dans un ordinateur.
- (3 points) En complément-2, quel est l'intervalle de valeurs pouvant être représentées sur 4 bits? Sur 8 bits? Indiquez la valeur minimale et maximale pour chaque cas. Dérivez ensuite une formule générale en fonction du nombre de bits N .
 - (4 points) Indiquez quel(s) drapeau(x) de l'ALU seront activés lors des deux opérations suivantes en complément-2 sur 4 bits: $6 + 2$ et $-6 - 2$.
 - (3 points) Un nombre entier non-signé sur 16 bits peut aussi être stocké en ASCII en représentant chaque chiffre par sa valeur ASCII correspondante. Donnez un exemple où stocker le nombre en ASCII requiert 1) *moins* de mémoire; 2) *autant* de mémoire; et 3) *plus* de mémoire que de le représenter en binaire. Vous devez donner un exemple pour chacune de ces situations.
5. (a) (1 point) Si un décodeur d'adresses utilise deux lignes du bus d'adresses pour le décodage, combien de composantes peuvent être contrôlés au maximum?
- (b) (2 points) Un bus sert à connecter plusieurs composantes ensemble. Comment faire pour s'assurer qu'une seule composante est active à la fois?
- (c) On veut accéder à une donnée située en mémoire RAM et la placer dans un registre du microprocesseur.
- (1 point) Quelle devraient être les valeurs sur le bus de contrôle, d'adresses, et de données?
 - (1 point) Laquelle de ces trois instructions devrait-on utiliser: LDR, STR, ou MOV?
6. (10 points) La table suivante montre différentes adresses de la mémoire d'un ordinateur ARM. À partir de cette table, et en assumant que la valeur initiale de PC soit 0x0, indiquez quelle sera la valeur des registres R0, R1, R2, R3 et PC *après* l'exécution de l'instruction B #0.

Note: l'annexe A pourrait vous aider à répondre à cette question.

Adresse	Instructions	Binaire
0x00000000	B #0x80	0xEA00001E
0x00000004	MUL R0, R7, R5	0xFFFFFFFF75
0x00000008	ANDEQ R0, R0, R2	0x000000E2
⋮	⋮	⋮
0x00000080	MOV R0, #4	0xE3A00004
0x00000084	LDR R1, [R0], #4	0xE5901000
0x00000088	LDR R2, [R0]	0xE5902004
0x0000008C	SUB R0, PC, R0	0xE04F0000
0x00000090	LSL R3, R2, #2	0xE0813002
0x00000094	B #-0x94	0xEAFFFFFE

7. Les questions suivantes font référence à l'architecture ARM.

- (a) (2 points) Décrivez le rôle du programme assembleur, comme celui que l'on retrouve dans IAR.
- (b) (2 points) Pourquoi ne peut-on pas référer à l'adresse d'une constante directement dans une instruction `MOV`?
- (c) (6 points) On vous donne le code assembleur suivant:

Adresse	Instructions
	<code>main</code>
<code>0x00000080</code>	<code>LDR R0, =maVariable</code>
<code>0x00000084</code>	<code>LDR R1, [R0]</code>
<code>0x00000088</code>	<code>MOV R1, #1</code>
<code>0x0000008C</code>	<code>ADD R0, R0, R1</code>
<code>0x00000090</code>	<code>B main</code>

L'assembleur remplacera l'instruction à l'adresse `0x80` car le microprocesseur ne comprend pas ce qu'est l'étiquette `maVariable`. En supposant que l'assembleur utilise l'adresse `0x94`, par quelle instruction l'assembleur remplacera-t-il l'instruction originale?

8. Répondez aux questions suivantes portant sur les interruptions.

- (a) (2 points) Quelle est la différence entre le traitement d'une interruption et un appel de fonction?
- (b) (2 points) Qu'est-ce qu'une exception? Donnez un exemple.
- (c) (2 points) Qu'est-ce que la table des vecteurs d'interruption? Quel programme la gère, habituellement?
- (d) (4 points) Décrivez brièvement ce qui se passe lorsqu'une interruption est soulevée.

9. (10 points) Une multiplication peut être représentée par une application successive d'additions. Par exemple, $3 \times 5 = 5 + 5 + 5 = 3 + 3 + 3 + 3 + 3 = 15$. Pour cette question, vous aurez à écrire une fonction en assembleur ARM qui implémente une multiplication grâce à une série d'additions.

Détails:

1. Ne traitez que le cas où les deux valeurs sont toujours strictement positives (> 0).
2. Votre fonction doit prendre ses arguments dans les registres R0 et R1, et doit retourner le résultat dans R0.
3. Votre fonction *ne doit pas* utiliser l'instruction MUL.
4. L'appel de votre fonction *ne doit pas* modifier les registres autres que R0.
5. L'annexe A contient une liste d'instructions et de codes de conditions pouvant vous être utiles.

main

```
MOV R0, #4
MOV R1, #8
BL maMultiplication ; appel de votre fonction, qui devrait calculer 4 x 8
MOV R3, R0          ; récupère valeur de retour, qui devrait contenir 32
...
```

maMultiplication

```
; écrivez le code de votre fonction.
```

Indices:

1. Additionnez la valeur contenue dans R0. Combien de fois faut-il l'additionner? Il faut le faire "R1-1" fois. Par exemple, plus haut nous avons écrit $3 \times 5 = 5 + 5 + 5$. Cela est donc deux additions: $(5 + 5) + 5$.
2. Testez votre fonction avec des valeurs simples (ex: 1×2 , 2×3) pour vous assurer de son bon fonctionnement.

Question bonus:

- (a) (5 points (bonus)) Modifiez votre code pour traiter correctement le cas où R0 ou R1 auraient une valeur de 0.

10. Indiquez si chacun des énoncés suivants est vrai ou faux.

- (a) (1 point) Selon la loi de Moore, si le nombre de transistors par pouce carré d'un microprocesseur typique était K en 2008, il était environ $8K$ en 2014.
- (b) (1 point) Un ordinateur ayant plusieurs mémoires aura plusieurs bus distincts pour accéder à chaque mémoire individuellement.
- (c) (1 point) Bien que la longueur d'une instruction peut varier d'un jeu d'instructions à un autre, toutes les instructions ont toujours la même taille à l'intérieur d'un jeu d'instructions.
- (d) (1 point) Le jeu d'instructions ARM est de type CISC.
- (e) (1 point) Pour diviser par deux en décalant des bits vers la droite, il faut que le bit le plus significatif du nombre décalé soit dupliqué.
- (f) (1 point) Une mémoire dynamique est une mémoire qui peut être lue seulement.
- (g) (1 point) Une mémoire est décrite avec deux nombres: la taille des mots, et le nombre d'adresses.
- (h) (1 point) Dans une architecture "Memory-Mapped I/O" (comme en ARM), on utilise des instructions différentes selon qu'on veuille accéder aux mémoires ou aux périphériques.
- (i) (1 point) Le décodeur d'adresse détermine quelle composante sera activée en fonction de la valeur placée sur le bus d'adresses.
- (j) (1 point) Le bus de données doit avoir la même taille que la largeur des mots stockés en mémoire.

A Annexe: Instructions ARM et codes de conditions

Dans la table suivante, Op1 dénote une opérande de type 1, et Op2 une opérande de type 2.

Mnémonique	Description
ADD Rd, Rs, Op1	$Rd = Rs + Op1$
ADC Rd, Rs, Op1	$Rd = Rs + Op1 + Carry$
AND Rd, Rs, Op1	$Rd = Rs \text{ AND } Op1$
ASR Rd, Rs, #imm	$Rd = Rs / 2^{imm}$
Bcc Offset	PC = PC + Offset, si cc est rencontré
BLcc Offset	Comme B, LR = Adr. de l'instr. 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	$Rd = Rs \times 2^{imm}$
MUL Rd, Rs, Op1	$Rd = Rs \times Op1$
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
SBC Rd, Rs, Op1	$Rd = Rs - Op1 - C$
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	$Rd = Rs - Op1$

Mnémonique	Condition	Mnémonique	Condition
CS	Carry Set	CC	Carry Clear
EQ	Equal (Zero Set)	NE	Not Equal (Zero Clear)
VS	Overflow Set	VC	Overflow Clear
GT	Greater Than	LT	Less Than
GE	Greater Than or Equal	LE	Less Than or Equal
PL	Plus (Positive)	MI	Minus (Negative)
HI	Higher Than	LO	Lower Than
HS	Higher or Same	LS	Lower or Same