

GIF-1001 Ordinateurs: Structure et Applications
Hiver 2017

Examen mi-session

28 février 2017

Durée: 170 minutes

Enseignant: Jean-François Lalonde

Cet examen comporte 7 questions sur 13 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'examen contient trois (3) annexes:
 - l'annexe A contient une liste d'instructions ARM ainsi des codes de conditions;
 - l'annexe B contient la table ASCII;
 - l'annexe C contient le jeu d'instructions du simulateur du TP1.

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

Question:	1	2	3	4	5	6	7	Total
Points:	15	15	10	10	20	20	10	100

Bonne chance!

1. (15 points) Répondez aux questions suivantes sur la représentation des données dans un ordinateur.
 - (a) (1 point) Combien de bits sont nécessaires pour stocker le nombre de secondes dans une minute?
 - (b) (3 points) Calculez $-5 - 5$ en complément-2 sur: 1) 4 bits; et 2) 5 bits. Dans les deux cas, montrez votre réponse en binaire et en décimal, et indiquez également s'il y a débordement.
 - (c) (2 points) Comment le micro-processeur fait-il pour savoir que 0x6162 signifie "ab" en ASCII plutôt que 24930 en décimal?
 - (d) Soit les deux nombres suivants en complément-2 sur 6 bits: 0x26 et 0x0F.
 - i. (1 point) Convertissez ces nombres en décimal.
 - ii. (2 points) Calculez la somme de ces deux nombres, et donnez la réponse en hexadécimal ainsi qu'en décimal.
 - (e) La norme IEEE754 encode des nombres rationnels sur 32 bits de la façon suivante:

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

et les bits sont stockés selon la figure 1:



Figure 1: Convention IEEE-754 sur 32 bits.

- i. (2 points) Quelle est la représentation de 12.25 en IEEE754 sur 32 bits?
 - ii. (2 points) Quelle est la représentation décimale de 0xC0D00000, encodé en IEEE754 sur 32 bits?
- (f) (2 points) En plus de la représentation sur 32 bits, la norme IEEE754 définit aussi une représentation sur 64 bits:

$$(\text{signe})1, \text{ mantisse} \times 2^{(\text{exposant}-1023)} .$$

Dans ce cas, 11 bits sont utilisés pour l'exposant, et 52 pour la mantisse, selon la figure 2:

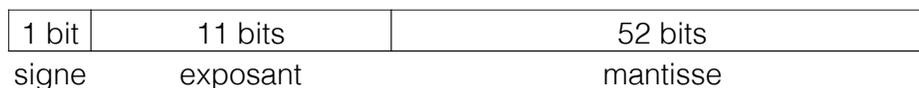


Figure 2: Convention IEEE-754 sur 64 bits.

Quelle est la représentation décimale de 0xC01A000000000000, encodé en IEEE754 sur 64 bits?

2. (15 points) Répondez aux questions suivantes portant sur le micro-processeur du simulateur du travail pratique 1. Dans ce système, toutes les instructions du microprocesseur sont encodées 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

Comme à l'habitude, le bit 0 est le moins significatif, et 15 le plus significatif. Le nombre identifiant le registre PC est 0xF (15), et le jeu d'instruction est décrit en annexe C.

- (a) (1 point) Combien d'instructions différentes ce système peut-il supporter? Expliquez pourquoi.
- (b) (1 point) Quelle est la différence entre les instructions MOV et LDR/STR?
- (c) (1 point) Dans l'instruction MOV R0, #0x70, est-il vrai que le < # > indique que 0x70 est une adresse? Pourquoi?
- (d) (5 points) Traduisez le programme suivant en binaire, et écrivez votre réponse en hexadécimal. Les numéros de ligne sont indiqués à gauche.

1	MOV R3, #0x24
2	ADD R2, R3
3	LDR R2, [R2]
4	STR R3, [R1]
5	JZE R0, R1

- (e) Soit 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	0x4300
2	0x1	0x4204
3	0x2	0x4040
4	0x3	0x8100
5	0x4	0x5001
6	0x5	0x1301
7	0x6	0x6201
8	0x7	0xF209
9	0x8	0x4F03
10	0x9	0x4040
11	0xA	0x9300

- i. (3 points) Écrivez le programme assembleur correspondant au code binaire ci-haut.
- ii. (4 points) Décrivez, *en une seule phrase*, ce que ce programme fait. Indiquez clairement les adresses employées pour les données en entrée et en sortie. *Indice*: pour déterminer ce que fait ce programme, placez de faibles valeurs fictives (e.g. entre 1 et 5) aux adresses mémoire 0x40 et subséquentes, exécutez ce programme pas à pas, et observez l'évolution du contenu des registres au fil du temps. *Important*: vous devez décrire le comportement global du programme; toute réponse décrivant les instructions une par une se verra attribuer la note de 0.

3. (10 points) Le schéma de la figure 3 représente l'architecture interne d'un microprocesseur simple, comme nous l'avons vu dans le cours. Utilisez ce schéma pour répondre aux questions suivantes.

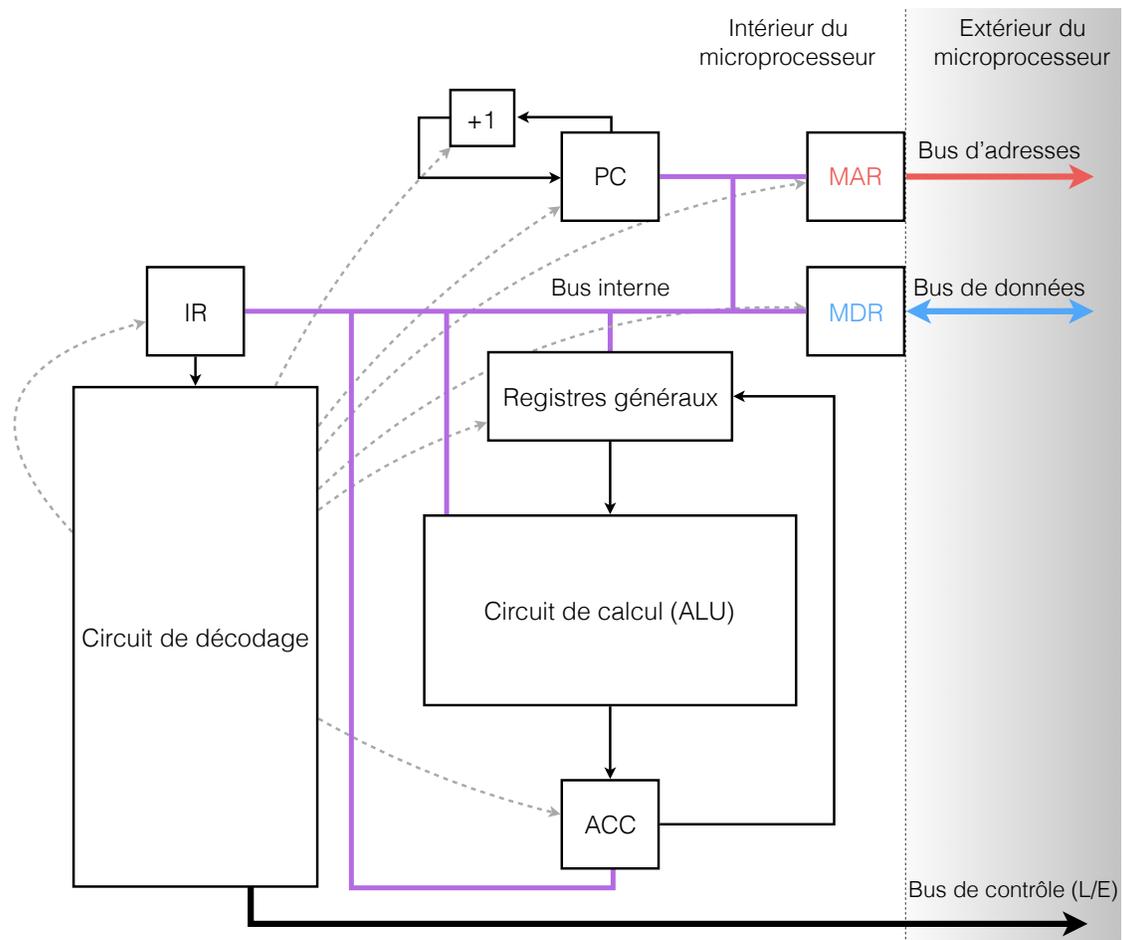


Figure 3: Architecture interne d'un microprocesseur simple.

- (a) (1 point) À quoi sert le registre IR?
- (b) (4 points) Décrivez les étapes qui doivent être effectuées lors d'un « *fetch* », soit lorsque l'instruction suivante est lue.
- (c) (2 points) À quoi servent les registres MAR et MDR?
- (d) (3 points) Écrivez les micro-instructions qui correspondent à l'exécution de l'instruction `SUB R1, #0x02` ($R1 \leftarrow R1 - 0x02$). Vous pouvez utiliser la notation « $Rd \leftarrow Rs$ » pour représenter un déplacement du contenu d'un registre source Rs vers un registre destination Rd , par exemple. N'oubliez pas d'indiquer les signaux du bus de contrôle, s'il y a lieu.

4. (10 points) Considérons une mémoire branchée sur un bus d'adresse de 12 bits et qui attribue une adresse à chaque octet. Une partie de son contenu est illustré ci-dessous.

Adresse	Valeur
⋮	⋮
0x050	0x00
0x051	0x61
0x052	0x42
0x053	0x63
0x054	0x21
0x055	0x00
0x056	0x6F
0x057	0x53
0x058	0x61
0x059	0x3F
⋮	⋮
0x2A0	0x23
0x2A1	0x67
0x2A2	0x69
0x2A3	0x66
⋮	⋮

- (a) (2 points) Quelle est la taille totale de la mémoire, en kilo-octets?
- (b) (2 points) Après avoir analysé un programme, vous avez déterminé qu'il met le mot de passe de l'utilisateur à l'adresse 86, et que le mot de passe a quatre (4) caractères encodés en ASCII. Quel est-il?
- (c) (2 points) En faisant l'hypothèse d'un système petit boutiste (« *little endian* ») et qu'un entier non-signé de 32 bits est stocké en mémoire à l'adresse 80, quelle est la valeur de cet entier en hexadécimal?
- (d) (2 points) En faisant l'hypothèse d'un système gros boutiste (« *big endian* ») et qu'un entier non-signé de 32 bits est stocké en mémoire à l'adresse 82, quelle est la valeur de cet entier en hexadécimal?
- (e) (2 points) Un programme charge une valeur de 32 bits entre les adresses 0x2A0 et 0x2A3. Est-ce que cette valeur est un nombre rationnel encodé en IEEE754 ou un entier non-signé?

5. (20 points) Répondez aux questions portant sur le code assembleur ARM suivant (les numéros de ligne sont indiqués à gauche):

```
1 SECTION INTVEC
2
3 B main
4 tableau DC32 0x10, 0x42, 0xA4, 0xA0, 0x32, 0x02, -1
5
6 SECTION CODE
7
8 main
9
10 LDR SP, =maPile
11 ADD SP, SP, #64
12
13 LDR R0, =tableau
14 BL fonctionMystere
15
16 B main
17
18 fonctionMystere
19 PUSH {R1, R2, R3, R5}
20 MOV R3, #0
21 LDR R1, [R0, R3]
22 MOV R2, R1
23
24 debut
25 CMP R1, #-1
26 BEQ fin
27
28 CMP R2, R1
29 MOVLT R2, R1
30 MOVLT R5, R3
31 ADD R3, R3, #4
32 LDR R1, [R0, R3]
33 B debut
34
35 fin
36 ASR R5, R5, #2
37 MOV R0, R5
38 POP {R1, R2, R3, R5}
39 BX LR
40
41 SECTION DATA
42
43 maPile DS32 16
```

- (a) (1 point) Pourquoi utilise-t-on l'instruction BL et non simplement B à la ligne 14?
- (b) (2 points) Pourquoi utilise-t-on les instructions PUSH et POP aux lignes 19 et 38?
- (c) (2 points) Pourquoi n'est-il pas nécessaire de mettre LR dans la liste des registres des instructions PUSH et POP aux lignes 19 et 38? Indiquez un scénario où il est nécessaire de le faire.
- (d) (2 points) Que fait l'instruction LDR R1, [R0, R3] aux lignes 21 et 32?

-
- (e) (2 points) Comment l'instruction `MOVLT R5, R3` fait-elle pour savoir si la condition "LT" est satisfaite? Quelle autre instruction affecte cette condition?
- (f) (1 point) À quelle opération mathématique l'instruction `ASR R5, R5, #2` à la ligne 36 correspond-elle?
- (g) (4 points) Quelle est la valeur de `R0` à la ligne 16?
- (h) Comme vous pouvez le constater, la fonction `fonctionMystere` parcourt les éléments du tableau `monTableau`.
- i. (1 point) Comment la fonction fait-elle pour savoir quand arrêter de boucler?
 - ii. (2 points) Quel registre est utilisé pour passer un argument à la fonction? À quoi cet argument correspond-il?
 - iii. (1 point) Quel registre est utilisé pour la valeur de retour de la fonction?
 - iv. (2 points) Décrivez *en une seule phrase* ce que fait cette fonction. *Important*: vous devez décrire le comportement global du programme; toute réponse décrivant les instructions une par une se verra attribuer la note de 0.

6. (20 points) Répondez aux questions suivantes, portant sur l'assembleur ARM.

- (a) (2 points) Nommez une caractéristique du jeu d'instruction ARM qui témoigne de sa nature RISC.
- (b) (2 points) Décrivez brièvement la principale fonction de chacun des registres suivants:
- i. PC
 - ii. LR
 - iii. SP
 - iv. CPSR
- (c) (5 points) Considérez le code suivant. Pour chaque ligne, on indique l'adresse (qui commence à 0x88), suivie de l'instruction en format binaire. Les numéros de ligne sont indiqués à gauche.

1	0x88	ADD R1, PC, #4
2	0x8C	PUSH {R1}
3	0x90	MOV R0, #0x88
4	0x94	MOV R1, #0x8A
5	0x98	ADD LR, PC, #12
6	0x9C	CMP R0, R1
7	0xA0	POPLT {R0}
8	0xA4	BXLT R0
9	0xA8	BX LR
10		fin
11	0xAC	B fin

Indiquez l'ordre des instructions exécutées par le microprocesseur en utilisant leur *numéro de ligne* correspondant. Vous pouvez assumer qu'une pile a préalablement été préparée. Une instruction conditionnelle (par exemple, MOVEQ) est considérée comme exécutée même si sa condition (par exemple, EQ) n'est pas satisfaite.

(d) (5 points) Écrivez du code assembleur qui branche à l'adresse:

- 0x80 si $R0 < 0$;
- 0x90 si $R0 = 0$;
- 0xA0 si $R0 > 0$.

(e) (6 points) Écrivez du code assembleur qui calcule la somme s des puissances de 3 entre 0 et N :

$$s = \sum_{n=0}^N 3^n,$$

où le résultat s est placé dans R1, et N dans R0. Implémentez le pseudo-code suivant:

R1	\leftarrow	0 ;
R2	\leftarrow	1 ;
while		$R0 \geq 0$ do
	R1	\leftarrow R1 + R2 ;
	R2	\leftarrow R2 \times 3 ;
	R0	\leftarrow R0 - 1 ;
end		

7. Répondez aux questions suivantes par une réponse courte.
- (a) (1 point) Vrai ou faux? Un pipeline à trois étages permet à un micro-processeur de lire, décoder, et exécuter la même instruction de façon simultanée.
 - (b) (1 point) Quelle est la taille des registres dans l'architecture ARM vue dans le cours?
 - (c) (1 point) Dans une architecture de type « memory-mapped I/O », de quelle façon détermine-t-on quel périphérique est activé?
 - (d) (1 point) Vrai ou faux? Lors de l'exécution d'une instruction LDR, le bus de contrôle est placé en lecture.
 - (e) (1 point) En utilisant l'assembleur ARM, donnez un exemple d'une ligne de code qui alloue en mémoire deux blocs de 32 bits chacun, sans toutefois leur attribuer de valeur.
 - (f) (1 point) Pourquoi faut-il incrémenter PC de 4 dans l'architecture ARM et non pas 1?
 - (g) (1 point) Vrai ou faux? Une pile est une structure de données de type « premier entré, premier sorti » (en anglais: FIFO, « first in, first out »)
 - (h) (1 point) Qu'est-ce qu'un ALU?
 - (i) (1 point) Quelles sont les trois grandes opérations du cycle d'instructions?
 - (j) (1 point) Combien de bits peut-on représenter avec un caractère hexadécimal?

A Annexe: Instructions ARM et codes de conditions

Instruction	Description
ADD Rd, Rs, Op1	$Rd \leftarrow Rs + Op1$
AND Rd, Rs, Op1	$Rd \leftarrow Rs \text{ AND } Op1$
ASR Rd, Rs, #imm	$Rd \leftarrow Rs / 2^{imm}$
B etiquette	$PC \leftarrow \text{adresse}(\text{etiquette})$
BL etiquette	$LR \leftarrow PC - 4, PC \leftarrow \text{adresse}(\text{etiquette})$
BX Rs	$PC \leftarrow Rs$
CMP Rs, Op1	Change les drapeaux comme $Rs - Op1$
LDR Rd, =etiquette	$Rd \leftarrow \text{adresse}(\text{etiquette})$
LDR Rd, [Rs, Op2]	$Rd \leftarrow \text{Mem}[Rs + Op2]$
LDR Rd, [Rs], Op2	$Rd \leftarrow \text{Mem}[Rs], Rs \leftarrow Rs + Op2$
LDR Rd, [Rs, Op2]!	$Rs \leftarrow Rs + Op2, Rd \leftarrow \text{Mem}[Rs]$
LSL Rd, Rs, #imm	$Rd \leftarrow Rs \times 2^{imm}$
MUL Rd, Rn, Rs	$Rd \leftarrow Rn \times Rs$
MVN Rd, Op1	$Rd \leftarrow !Op1$ (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]	$\text{Mem}[Rs + Op2] \leftarrow Rd$
STR Rd, [Rs], Op2	$\text{Mem}[Rs] \leftarrow Rd, Rs \leftarrow Rs + Op2$
STR Rd, [Rs, Op2]!	$Rs \leftarrow Rs + Op2, \text{Mem}[Rs] \leftarrow Rd$
SUB Rd, Rs, Op1	$Rd \leftarrow Rs - Op1$

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.

B 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	i	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				

C Annexe: Jeu d'instructions du microprocesseur du TP1

Mnémonique	Opcode	Description
MOV Rd, Rs	0000	$Rd \leftarrow Rs$
MOV Rd, Const	0100	$Rd \leftarrow \text{Const}$
ADD Rd, Rs	0001	$Rd \leftarrow Rd + Rs$
ADD Rd, Const	0101	$Rd \leftarrow Rd + \text{Const}$
SUB Rd, Rs	0010	$Rd \leftarrow Rd - Rs$
SUB Rd, Const	0110	$Rd \leftarrow Rd - \text{Const}$
LDR Rd, [Rs]	1000	$Rd \leftarrow \text{Mem}[Rs]$
STR Rd, [Rs]	1001	$\text{Mem}[Rs] \leftarrow Rd$
JZE Rc, Const	1111	si $Rc = 0$, $PC \leftarrow \text{Const}$
JZE Rc, Rs	1011	si $Rc = 0$, $PC \leftarrow Rs$

Table 3: Jeu d'instructions du microprocesseur du TP1