GIF-1001 Ordinateurs: Structure et Applications
Automne 2017
Examen mi-session
26 octobre 2017
Durée: 150 minutes

Cet examen comporte 6 questions sur 12 pages (incluant celle-ci). L'examen compte pour 40% de la note totale pour la session. Assurez-vous d'avoir toutes les pages.
Tout objet ne faisant pas partie des listes de matériel obligatoire ou autorisé qui est utilisé ou présent sur votre table de travail est considéré comme de la tricherie.

Liste de matériel obligatoire:

- Carte étudiante;
- Crayon plomb, stylo ou plume.

Liste de matériel autorisé facultatif:

- Une feuille aide-mémoire 8.5×11 recto-verso écrite à la main;
- Deux crayons plomb, pousses-mines, stylos ou plumes;
- Deux gommes à effacer et/ou ruban correcteur;
- Un aiguise crayon;
- Trois surligneurs;
- Une règle de 15 ou 30 cm, un rapporteur d'angles et un compas;
- Deux feuilles brouillon blanches 8.5×11;
- Deux boissons ou collations (eau, café, barre tendre, sac de beef jerky, etc.);
- Une paire de bouchons d'oreilles.

Instructions:

- Placez votre carte étudiante bien visible sur votre table de travail;
- Écrivez votre nom et votre NI sur le cahier de réponses qui vous a été remis;
- Écrivez vos réponses dans le cahier de réponses qui vous a été remis;
- L'annexe A contient une liste d'instructions ARM;
- L'annexe B contient la table ASCII.

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

Question:	1	2	3	4	5	6	Total
Points:	6	15	11	11	14	10	67

Bonne chance!

1. Répondez aux questions suivantes sur les ordinateurs.
   (a) (1 point) Quel est le nom de l'observation selon laquelle le nombre de transistors dans les circuits intégrés double environ au deux ans?

   Solution: La loi de Moore.

   (b) (2 points) Selon l'architecture de von Neumann, quelles sont les quatre (4) composantes principales d'un processeur?

   Solution: La mémoire, l'ALU, le CCU et l'équipement d'entrées-sorties. (0,5 pts par élément)

   (c) (1 point) Dans le cycle d'instruction, que fait l'opération Fetch?

   Solution: Lis l'instruction en mémoire.

   (d) (2 points) Une clef USB est-elle une entrée ou une sortie pour un ordinateur? Justifiez votre réponse.

   Solution: C'est à la fois une entrée et une sortie car l'ordinateur peut lire des données de la clef USB (entrée) et aussi y en écrire (sortie).
   1 pt pour entrée+sortie 1 pt pour justification

2. Répondez aux questions suivantes sur la représentation des données dans un ordinateur.
   (a) (1 point) Combien de bits sont nécessaires pour stocker le nombre entier de degrés dans un cercle?

   Solution: Neuf (9) bits permettent de stocker 512 valeurs, c'est donc suffisant.

   (b) (2 points) Convertissez le nombre décimal 543 en hexadécimal sur 16 bits.

   Solution: 543 = 0x021F

   (c) (2 points) Additionnez les deux nombres suivants qui sont donnés en représentation complément-2 sur 20 bits et donnez la réponse en hexadécimal et en décimal: 0xFFFF8 et 0xFFFE2.

   Solution: 0xFFFDA, -38.

   (d) (2 points) Lors d'une addition de deux nombres en complément-2, comment peut-on faire pour détecter un débordement? Donnez un exemple sur 4 bits.

   Solution: On peut détecter un débordement lorsque le bit de signe est le même pour les deux opérandes en entrée, et qu'il change en sortie. Par exemple, $5 + 6$ est, en binaire, $0101_b + 0110_b = 1011_b$, soit -5. (2 pts si l'explication est entièrement correcte)

(e) (2 points) Donnez le nombre de bits à 1, le nombre de bits à 0 et le nombre total de bits dans le nombre suivant: 0x569A00FF.

Solution: 32 bits au total dont 16 à 0 et 16 à 1. (-1 si un des trois n'est pas correct, 0 si 2 erreurs)

(f) (2 points) Que représente 0x3FA00000 encodé 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.

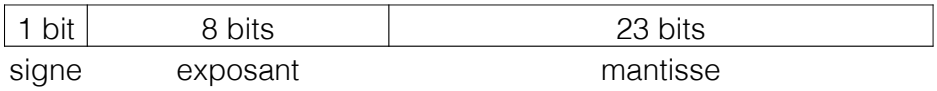

Figure 1: Convention IEEE-754 sur 32 bits.

Solution: 1.25

(g) (2 points) Que représente 0x7F800000 encodé sur 32 bits avec IEEE 754?

Solution: ∞ (l'infini).

(h) (2 points) Que représente 0x7FE00000 encodée sur 32 bits avec IEEE 754?

Solution: Ce n'est pas un nombre, not a number ou NaN.

3. Pour les questions suivantes, considérez une mémoire pouvant stocker 1024 octets (1 octet par adresse) dont une partie du contenu est illustré ci-dessous.

Adresse	Valeur		Adresse	Valeur
...	...		...	...
60_h	00_h		70_h	00_h
61_h	61_h		71_h	61_h
62_h	23_h		72_h	42_h
63_h	44_h		73_h	63_h
64_h	61_h		74_h	21_h
65_h	45_h		75_h	00_h
66_h	$6E_h$		76_h	10_h
67_h	65_h		77_h	12_h
68_h	72_h		78_h	14_h
69_h	79_h		79_h	15_h
$6A_h$	73_h		$7A_h$	23_h
$6B_h$	$7A_h$		$7B_h$	67_h
$6C_h$	$6B_h$		$7C_h$	69_h
$6D_h$	66_h		$7D_h$	68_h
$6E_h$	$4E_h$		$7E_h$	62_h
$6F_h$	50_h		$7F_h$	63_h
...	...		...	...

(a) (1 point) Quelle est la largeur minimale requise du bus d'adresse pour accéder à toutes les adresses de cette mémoire?

Solution: 10 bits.

(b) (1 point) Quelle est la largeur du bus de données utilisé par cette mémoire?

Solution: 8 bits.

(c) (2 points) Donnez le nombre total de bits pouvant être stockés dans cette mémoire.

Solution: 8192 bits.

(d) (3 points) Après avoir analysé un programme, vous avez déterminé qu'il met le mot de passe de l'utilisateur à l'adresse 98, et que le mot de passe a neuf (9) caractères. Quel est-il?

Solution: `#DaEnerys` (-1 pt par caractère fautif)

(e) (2 points) En faisant l'hypothèse d'un système petit boutiste (*little endian*) et qu'un nombre non-signé de 32 bits est en mémoire à l'adresse 0x74, quel est ce nombre en hexadécimal?

Solution: 0x12100021

(f) (2 points) Quel type d'information vu dans le cadre du cours (entier, rationnel, caractère, chaîne de caractères, ...) est stocké en mémoire aux adresses $7A_h$ à $7D_h$ inclusivement?

Solution: On ne peut pas savoir. Peut-être une chaîne de caractères, peut-être un nombre entier 32 bits, peut-être 2 nombres entiers 16 bits, etc.

4. Répondez aux questions suivantes portant sur le micro-processeur du simulateur du travail pratique 1. Un rappel du jeu d'instructions est fourni:

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 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).

Table 1: Jeu d'instructions du microprocesseur

(a) (2 points) Traduisez l'instruction suivante en hexadécimal:

```
1 MOV R3 #35
```

Solution:
`0x4323`

(b) (2 points) Quelle instruction est représenté par l'hexadécimal suivant:
`0x8203`

Solution:
`LDR R2 [R3]`

(c) (7 points) Soit le programme suivant. Pour chaque ligne, on indique l'adresse (qui commence à `0x0`), suivie de l'instruction. Décrivez, en une seule phrase, ce que ce programme fait. Indiquez clairement les adresses employées pour les données en entrée et en sortie. Important: vous devez décrire le comportement global du programme; toute réponse décrivant les instructions une par une se verra attribuer la note de 0.

```
1 0x0   MOV R0 #2
2 0x1   LDR R0 [R0]
3 0x2   MOV R2 #3
4 0x3   MOV R3 #3
```

```
0x4   ADD R2 R3
0x5   ADD R0 R0
0x6   SUB R2 #1
0x7   JZE R2 #5
0x8   STR R0 [R2]
```

Solution: Il charge la valeur en mémoire à l'adresse 0x2, la multiplie par 2, puis l'écrit en mémoire à l'adresse 0x5. (1 point pour charge la valeur à l'adresse 0x2, 3 point pour la multiplie par 2, 1 point pour écrit le résultat à l'adresse 0x5)

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

   (a) (1 point) Auquel des deux grands types de jeu d'instructions le jeu d'instructions ARM appartient-il?

   Solution: RISC.

   (b) (3 points) Nommez trois registres spéciaux utilisés dans l'architecture ARM et leurs fonctions.

   Solution:

   - PC (Program counter) Adresse de l'instruction à lire
   - LP (Link Register) Adresse de retour de fonction
   - SP (Stack Pointer) Adresse du sommet de la pile

   (0,5 par registre et par fonction)

   (c) (5 points) Écrivez du code assembleur qui sautera les 9 instructions qui se trouveront après votre code, peu importe lesquelles, si le nombre contenu dans R4 est négatif. Votre code doit fonctionner peu importe où il se situe en mémoire.

   Solution: Par exemple:

```
CMP R4, #0
ADDLT PC, PC, #32
```

   (5 points pour une solution fonctionnelle)

   (d) (5 points) Écrivez du code assembleur qui est une fonction appelée par le code ci-dessous et qui place le double du nombre contenu dans R1 à l'adresse 42 en mémoire. Votre fonction doit s'assurer que les registres ne soient pas modifiés pour l'appelant. Considérez qu'une pile a déjà été préparée.

```
1 BL uwotm8
```

   Solution: Par exemple:

```
uwotm8
  PUSH {R0, R1}

  ADD R1, R1, R1
  MOV R0, #42
  STR R1 [R0]

  POP {R0, R1}
  BX LR
```

   1 pt étiquette 1 pt push-pop 2 pt mettre R1*2 à l'adresse 42 1 pt BX LR

6. Répondez aux questions suivantes.

   (a) (1 point) Donnez un exemple d'un nombre hexadécimal qu'on ne peut pas représenter en décimal.

   Solution: Il n'y en a pas.

   (b) (1 point) Vrai ou faux? Lors de l'exécution d'une instruction `STR`, le bus de contrôle est placé en lecture.

   Solution: Faux

   (c) (1 point) En utilisant l'assembleur ARM, donnez un exemple d'une ligne de code qui réserve en mémoire quatre (4) blocs de 32 bits chacun, sans toutefois leur attribuer de valeur.

   Solution: `etiquette DS32 4`

   (d) (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 »)

   Solution: Faux, elle est de type LIFO, « last in, first out ».

   (e) (1 point) Combien de bits peut-on représenter avec un caractère hexadécimal?

   Solution: 4 bits

   (f) (1 point) Donnez deux exemples de mnémoniques d'instructions de déplacement de données dans le jeu d'instruction de la plateforme ARM7TDMI.

   Solution: MOV, LDR.

   (g) (1 point) En assembleur ARM, donnez l'instruction qui permet de placer l'adresse de `maVariable` dans le registre R10.

   Solution: `LDR R10, =maVariable`

   (h) (1 point) Dans une architecture de type "memory-mapped I/O", quelle composante est responsable d'activer la broche d'activation (*enable*) des périphériques?

   Solution: Le décodeur d'adresse.

   (i) (1 point) Expliquez la différence entre les mémoires volatiles et non-volatiles?

   Solution: Les mémoire volatiles perdent leur contenu lorsqu'elles perdent leur alimentation.

   (j) (1 point) Pourquoi faut-il incrémenter PC de 4 dans l'architecture ARM et non pas 1?

Solution: Car chaque instruction est encodée sur 4 octets, et que chaque octet possède une adresse différente.

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}`	Récupère la liste de registres de 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 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.

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	¡	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				