GIF-1001 Ordinateurs: Structure et Applications
Hiver 2024
Examen mi-session
14 mars 2024
Durée : 170 minutes
Professeur : Yannick Hold-Geoffroy

Cet examen comporte 7 questions sur 11 pages (incluant celle-ci et les annexes), comptabilisées sur un total de 100 points. L'examen compte pour 40% de la note totale pour la session.

- Vous avez droit à une calculatrice acceptée ;
- Assurez-vous que l'étiquette sur le cahier de réponse corresponde bien à vous ;
- Assurez-vous d'avoir toutes les pages ;
- **Certaines questions apparaissent au verso** : regardez des deux côtés des pages !
- Écrivez vos réponses dans le cahier bleu qui vous a été remis ;
- SVP sortez vos cartes étudiantes et placez-la visiblement sur votre table de travail ;
- L'examen contient quatre (4) annexes :
  - l'annexe A contient un rappel sur les unités en binaire et les logarithmes ;
  - l'annexe B contient le jeu d'instructions du simulateur du TP1 ;
  - l'annexe C contient la table des valeurs hexadécimales ;
  - l'annexe D contient une liste d'instructions ARM ainsi que des codes de conditions ;
  - l'annexe E contient la table ASCII.

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

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

Bonne chance !

1. (15 points) Répondez aux questions suivantes sur la représentation des données dans un ordinateur. **N'oubliez pas les préfixes 0b pour les réponses en binaire et 0x pour les réponses en hexadécimal.**
   (a) (1 point) Combien de bits sont nécessaires pour stocker le nombre de jours entiers dans une année du calendrier occidental ? (Rappel : il y a un maximum de 366 jours entiers dans le calendrier occidental.)
   (b) (2 points) Soit la valeur 0xF28EA322.
      i. Cette valeur, telle que représentée, est encodée sur combien de bits ?
      ii. Que signifie cette valeur si elle est observée dans la mémoire d'un ordinateur ? Vous n'avez pas accès au code qui a défini cette valeur.
   (c) (1 point) Convertissez la valeur décimale -5 en binaire sur 4 bits avec la représentation complément-2 (représentation signée).
   (d) (4 points) Calculez le résultat de $3 + 6$ en complément-2 (représentation signée) :
      i. Sur 4 bits. Écrivez votre réponse en binaire et en décimal. Indiquez s'il y a débordement.
      ii. Sur 5 bits. Écrivez votre réponse en binaire et en décimal. Indiquez s'il y a débordement.
   (e) (6 points) La norme IEEE 754 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 :

1 bit	8 bits	23 bits
signe	exposant	mantisse

FIGURE 1 – Convention IEEE-754 sur 32 bits.

      i. Quelle est la représentation de -7.25 en IEEE754 sur 32 bits ? Écrivez votre résultat en hexadécimal.
      ii. Quelle est la représentation décimale de 0x40480000, encodé en IEEE754 sur 32 bits ?
   (f) (1 point) Est-ce que la multiplication des deux nombres 0xC34D2000 et 0x436EC000 représentés en IEEE754 donnera un résultat plus grand ou plus petit que 0 ?

2. (10 points) Répondez aux questions suivantes portant sur le microprocesseur 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 B.
   (a) (2 points) Le jeu d'instruction de ce système comprend 10 instructions décrites en annexe. Serait-il possible d'avoir plus d'instruction sans changer le système ?
   Si oui, combien d'instructions additionnelles sont possibles, et pourquoi ?
   Si non, décrivez les modifications à apporter pour pouvoir ajouter une nouvelle instruction.
   (b) (2 points) En vous référant au jeu d'instruction en annexe B, est-il possible d'employer une instruction `MOV` pour effectuer un accès dans la mémoire ?
   (c) Soit le programme suivant en assembleur du TP1 (voir annexe B). Pour chaque ligne, on indique l'adresse (qui commence à `0x0`), suivie de l'instruction en format binaire entre parenthèses et de certaines de ces instructions décodées en texte. Les numéros de ligne sont indiqués à gauche.

```
1  0x0    (0x4040)    MOV R0, #0x40
2  0x1    (0x8000)    LDR R0, [R0]
3  0x2    (0x4104)    MOV R1, #0x04
4  0x3    (0x4200)    MOV R2, #0x00
5  0x4    (0xF108)    ???
6  0x5    (0x1200)    ???
7  0x6    (0x6101)    SUB R1, #0x01
8  0x7    (0x4F04)    ???
9  0x8    (0x4141)    MOV R1, #0x41
10 0x9    (0x9201)    STR R2, [R1]
```

      i. (3 points) Les instructions en texte aux lignes 5, 6 et 8 ont été perdues. Décodez-les en instructions assembleur à partir de l'instruction en format binaire entre parenthèses.
      ii. (3 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) à l'adresse mémoire 0x40, exécutez ce programme une instruction à la fois, et observez l'évolution du contenu des registres au fil du temps. Dans votre réponse, indiquez : 1) l'entrée du programme, 2) l'opération qui est effectuée, et 3) la sortie du programme. *Important* : vous devez décrire le comportement global du programme ; toute réponse décrivant les instructions une par une recevra la note de 0.

3. (15 points) 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 24 bits ;
   — une mémoire RAM où chaque octet possède une adresse différente ;
   — deux autres périphériques sont branchés sur les bus ;
   — il stocke les données en mémoire avec la convention petit boutiste («little endian») ;
   — si on nomme les bits les plus significatifs (MSB) du bus d'adresse b_{15} et b_{14}, le décodeur sélectionne les périphériques de la façon suivante :

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

   Ici, «X» indique soit 0 ou 1, sans importance.

   (a) (2 points) Quelle est la taille maximale de la mémoire RAM ? Écrivez votre réponse en kilo-octets (Ko), et écrivez votre démarche. N'oubliez pas que 1 Ko = 1024 octets.

   (b) On emploie une instruction pour enregistrer la valeur 0xBACCA1 à l'adresse 0x403A.
      i. (1 point) À quel périphérique cette addresse correspond-elle ?
      ii. (2 points) Indiquez les adresses dans le périphérique correspondant qui sont modifiées par cette instruction, ainsi que la valeur placée à chacune de ces adresses.

   (c) (4 points) La carte de la mémoire («memory map») d'un système indique les premières et dernières adresses du microprocesseur correspondant à chaque périphérique branché sur les bus. Quelle est la carte de la mémoire de ce système ?

   (d) (2 points) Pourrions-nous ajouter un périphérique à ce système sans changer le bus d'adresse ni son décodeur ? Pourquoi ?

   (e) On désire ajouter deux nouveaux périphériques à ce système.
      i. (2 points) Faut-il changer quelque chose au bus données ? Si oui, expliquez l'impact de ce changement.
      ii. (2 points) Faut-il changer quelque chose au bus d'adresse ? Si oui, expliquez l'impact de ce changement.

4. (15 points) Considérons une mémoire branchée sur un bus d'adresse de 14 bits et qui attribue une adresse à chaque octet. Une partie de son contenu est illustré ci-dessous.

Adresse	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
⋮	⋮															
0x140	01	12	34	56	79	89	AB	CD	EE	FF	50	43	2B	34	00	D2
0x150	01	0A	A0	E3	00	10	90	E5	04	20	90	E5	02	30	90	E7
⋮	⋮															
0x480	01	41	90	E7	01	00	80	E2	FE	FF	FF	EA	1E	00	00	EA
⋮	⋮															

   (a) (2 points) Quelle est la taille totale de la mémoire en kilo-octets (Ko) ?

   (b) (4 points) Il y a une chaîne de caractère de quatre (4) octets encodée en ASCII à l'adresse 330. Quelle est cette chaîne de caractère ? (Indice : employez la table ASCII à l'annexe E.)

   (c) (3 points) En faisant l'hypothèse d'un système petit boutiste (« *little endian* ») et qu'un entier non-signé de 16 bits est stocké en mémoire à l'adresse 322, quelle est la valeur de cet entier en hexadécimal ?

   (d) (3 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 0x154, quelle est la valeur de cet entier en hexadécimal ?

   (e) (1 point) Un programme charge une valeur de 32 bits entre les adresses 0x480 et 0x484 inclusivement. Est-ce que cette valeur est un nombre rationnel encodé en IEEE754 ou un entier non-signé ?

   (f) (2 points) Vous apprennez qu'il y a une instruction ARM encodée sur 32 bits à l'adresse `0x484`, il s'agit de `ADD R0, R0, #1`. Sachant que les bits b_{24} à b_{21} d'une instruction ARM correspond à son opcode, quel est l'opcode de ce format de l'instruction `ADD` ? Ici, b_0 est le bit le moins significatif et b_{31} le plus significatif.

5. (15 points) Répondez aux questions portant sur le code assembleur ARM suivant (les numéros de ligne sont indiqués à gauche) :

```
1  SECTION INTVEC
2  B main
3
4  SECTION CODE
5  main
6  LDR SP, =maPile
7  ADD SP, SP, #8
8  LDR R0, =donnees
9  BL fonctionMystere
10 B main
11
12 fonctionMystere
13 PUSH {R1, R2}
14 MOV R2, #0x7FFFFFFF
15 debut
16 LDR R1, [R0], #4
17 CMP R1, #0x80000000
18 BEQ fin
19 CMP R1, R2
20 MOVLT R2, R1
21 B debut
22
23 fin
24 MOV R0, R2
25 POP {R1, R2}
26 BX LR
27
28 SECTION DATA
29 donnees    ASSIGN32 0xA0, 0x08, 0x42, 0x8BCD, 0x80000000, 0x0
30 maPile     ALLOC32  2
```

(a) (2 points) Pourquoi doit-on rajouter 8 à `SP` à la ligne 7 ? (Rappel : SP est le pointeur de pile, une structure « Last-In, First-Out. »)

(b) (2 points) Pourrions-nous déclarer une variable avant la ligne 2, soit avant l'instruction `B main`, sans changement à l'exécution de ce programme ?

(c) (1 point) Pourquoi utilise-t-on l'instruction BL et non simplement B à la ligne 9 ?

(d) (2 points) Comment l'instruction `BEQ fin` (ligne 18) fait-elle pour savoir si la condition `EQ` est satisfaite ? Quelle autre instruction affecte cette condition ?

(e) La fonction `fonctionMystere` parcourt les éléments du tableau `donnees` avec la boucle débutant par l'étiquette `debut`.

i. (2 points) Quel est le critère d'arrêt de la fonction qui la fait arrêter de boucler ?

ii. (2 points) Quel registre est employé pour passer le paramètre d'entrée de la fonction ? À quoi cette valeur correspond-elle ?

iii. (1 point) Quel registre est employé pour passer la valeur de sortie de la fonction ?

iv. (3 points) Décrivez *en une seule phrase* ce que fait `fonctionMystere`. *Important* : vous devez décrire le comportement global du programme ; toute réponse décrivant les instructions une par une recevra la note de 0.

6. (15 points) Répondez aux questions suivantes, portant sur l'assembleur ARM.
   (a) (2 points) Pourquoi le processeur incrémente-t-il `PC` de 4 à chaque instruction ?
   (b) (1 point) Expliquez la différence entre « big endian » et « little endian ».
   (c) (4 points) Écrivez un court programme en assembleur ARM qui effectue les trois étapes suivantes :
      1. Multiplie la valeur contenue dans `R0` (il s'agit d'un entier) par 4, *sans utiliser l'instruction `MUL`*, et stocke le résultat dans `R1` ;
      2. Si le résultat de la multiplication est plus grand ou égal à 16, brancher à l'adresse `0x120` ;
      3. Si le résultat de la multiplication est plus petit que 16, brancher à l'adresse `0x90`.
   (d) Soit le code suivant. Pour chaque ligne, on indique l'adresse (qui commence à `0x80`), suivie de l'instruction. Les numéros de ligne sont indiqués à gauche.

```
1 0x80   MOV R0, #0x55
2 0x84   MOV R2, #0
3 0x88   AND R1, R0, #1
4 0x8C   CMP R1, #0
5 0x90   ADDNE R2, R2, #1
6 0x94   LSR R0, R0, #1
7 0x98   CMP R0, #0
8 0x9C   MOVNE PC, #0x88
9 0x100  MOV R0, R2
```

      i. (5 points) Indiquez l'ordre des 21 premières instructions exécutées par le microprocesseur en utilisant leur *numéro de ligne* correspondant. Vous devez indiquer la ligne d'une instruction conditionnelle (par exemple, `ADDNE`) même si sa condition (par exemple, `NE`) n'est pas satisfaite.
      ii. (3 points) Décrivez *en une seule phrase* ce que fait ce programme. Ne décrivez pas les instructions une-à-une. Assumez que `R0` (employé à la ligne 1) est la variable d'entrée du programme et peut être changée par un utilisateur.

7. (15 points) Répondez aux questions suivantes par une réponse courte.
   (a) (1 point) Vrai ou faux ? L'hexadécimal est une façon plus compacte de représenter du binaire.
   (b) (2 points) Quel est l'avantage d'employer un pipeline à trois étages (comme l'ARM du TP2) plutôt qu'un système sans pipeline ?
   (c) (1 point) Combien de bits peut-on représenter avec deux caractère hexadécimaux, comme ceux-ci : `0xFF` ?
   (d) (1 point) Dans un système de type « memory-mapped I/O » à plusieurs périphériques, quelle composante permet au processeur de sélectionner un périphérique spécifique ?
   (e) (1 point) Vrai ou faux ? Lors de l'exécution d'une instruction `STR`, le bus de contrôle est placé en lecture.
   (f) (1 point) Quelles sont les deux étapes effectuées par un microprocesseur ARM lorsqu'il exécute l'instruction `BL etiquette` ?
   (g) (1 point) Vrai ou faux ? Les trois principaux bus sont les bus d'adresses, de données et de contrôle.
   (h) (2 points) Quelles sont les trois étapes principales du cycle d'instructions ?
   (i) (1 point) Vrai ou faux ? Un débordement (« overflow » en anglais) signifie qu'une opération arithmétique sur deux entiers non-signés est invalide.
   (j) (2 points) Dans l'architecture ARM employée dans le travail pratique 2, pourquoi lorsqu'on lit le registre du « Program Counter » (`PC`), la valeur pointe deux instructions (+8 octets) plus loin ?
   (k) (2 points) On demande à un ordinateur d'effectuer l'opération `a` $\leftarrow$ `a + b`, où `a` et `b` sont des variables représentées en IEEE 754 avec une précision simple sur 32 bits (*float*). Les valeurs initiales de `a` et `b` sont respectivement 1.0 et 1×10^{-12}. Quelle sera la valeur de `a` après avoir exécuté cette opération 1 milliard (1 000 000 000) de fois ? (Indice : évaluez la taille de la mantisse sur la figure 1.)

A Annexe : Unités et logarithmes

A.1 Unités

Petit rappel sur les unités :

$$\begin{array}{llll} 1\text{Ko} & = 2^{10}\text{o} & = & 1\,024 \text{ octets} \\ 1\text{Mo} & = 2^{20}\text{o} = 1\,024\text{Ko} & = & 1\,048\,576 \text{ octets} \\ 1\text{Go} & = 2^{30}\text{o} = 1\,024\text{Mo} & = & 1\,073\,741\,824 \text{ octets} \end{array}$$

A.2 Logarithme en base 2

On peut calculer des logarithmes en base 2 à partir de logarithmes dans une autre base N (ex : 10). Pour ce faire, appliquez l'équation suivante :

$$\log_2 x = \frac{\log_N x}{\log_N 2}\,.$$

B Annexe : Jeu d'instructions du microprocesseur du TP1

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

TABLE 1 – Jeu d'instructions du microprocesseur du TP1

C Annexe : Table hexadécimale

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadécimal (0x)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Binaire (0b)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

D 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` / $2^{\texttt{cte}}$ (arithmétique, entier signé)
`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` × $2^{\texttt{cte}}$
`LSR Rd, Rb, #cte`	`Rd` ← `Rb` / $2^{\texttt{cte}}$ (logique, entier non-signé)
`MOV Rd, Op1`	`Rd` ← `Op1`
`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 2 – 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 3 – Codes de condition.

E 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				