UNIVERSITE Joseph FOURIER, Grenoble
Licence de Sciences et technologie - Parcours MIN, BIN, INF, IAG

Contrôle continu UE INF241 : Introduction aux Architectures Logicielles et Matérielles

9 Mars 2015, durée 1 h 30
Documents, calculettes, téléphones portables interdits
Le barême est donné à titre indicatif

1 Numération (5 points)

Pour cet exercice, **toutes les réponses devront être justifiées** (poser chaque opération avec les opérandes, les retenues, le résultat apparent et les indicateurs).

(a) Poser chacune des additions binaires suivantes : $(+124)_{10} + (-33)_{10}$ et $(+124)_{10} + (+38)_{10}$,
   1. **(1 point)** pour le résultat apparent sur 8 bits (codage en complément à deux),
   2. **(1 point)** pour les indicateurs Z, N, C et V.

(b) Poser chacune des soustractions binaires suivantes : $(+124)_{10} - (+38)_{10}$ et $(-33)_{10} - (+124)_{10}$,
   1. **(1 point)** pour le résultat apparent sur 8 bits (codage en complément à deux),
   2. **(1 point)** pour les indicateurs Z, N, C et V.

   Les soustractions seront réalisées par addition du complémentaire.

(c) **(1 point)** En supposant une mémoire d'octets avec des adresses de 32 bits, quelle est la taille maximale de la mémoire ?

2 Codage d'une instruction en ARM (3 points)

Une instruction de calcul `ARM` entre une valeur stockée dans un registre et une valeur immédiate sur 8 bits est encodée selon le schéma suivant (figure 1) :

— `cond` correspond au code de la condition (expression booléenne des indicateurs Z, N, C et V) à vérifier pour exécuter l'instruction,
— `I` doit être positionné à 1 pour signifier que le second opérande est une valeur immédiate,
— `code-op` encode la nature de l'opération (mnénomique de l'instruction : par exemple, le code de `ADD`),
— `S` vaut 1 si on souhaite mettre à jour les indicateurs avec le résultat de l'opération,
— `rn` est le numéro du registre premier opérande,
— `rd` est le numéro du registre où est stocké le résultat, et
— `immediate-8` est la valeur immédiate (second opérande) sur 8 bits.

Les codes des conditions et des opérations sont donnés dans les annexes A et B.

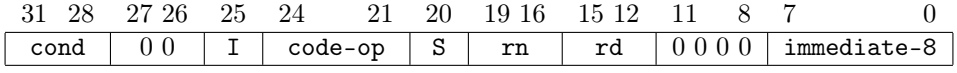

FIGURE 1 – Codage d'une instruction arithmétique ou logique

Questions :

(d) Donner le code de l'instruction `add r1,r8,#28` en binaire. **(2 points)**

(e) Donner le code de l'instruction `add r1,r8,#28` en hexadécimal. **(1 point)**

3 Programmation en ARM

3.1 Code Mystère (7 points)

Nous considérons le code suivant :

```
 1          .data
 2 CH:      .asciz "resultat : "
 3          .balign 2
 4 D1:      .hword 12
 5 F1:      .hword 56
 6 D2:      .hword 5
 7 F2:      .hword 25

 8          .text
 9          .global main
10 main:
11          ldr r0,ptrD1
12          ldrh r2,[r0]
13          ldr r0,ptrF1
14          ldrh r3,[r0]
15          ldr r0,ptrD2
16          ldrh r4,[r0]
17          ldr r0,ptrF2
18          ldrh r5,[r0]
19          cmp r2,r4
20          movhi r2,r4      @ mov, condition hi
21          cmp r3,r5
22          movlo r3,r5      @ mov, condition lo
23          ldr r1,ptrCH
24          bl EcrChaine
25          mov r1,r2
26          bl EcrNdecimal16
27          mov r1,r3
28          bl EcrNdecimal16
29          bal exit
30 ptrD1:   .word D1
31 ptrF1:   .word F1
32 ptrD2:   .word D2
33 ptrF2:   .word F2
34 ptrCH:   .word CH
```

Questions (justifiez brièvement votre raisonnement) :

(f) Rappelez la signification de `.balign 2`. **(0,5 point)**

(g) En supposant que le segment `.data` débute à l'adresse `0x100C`, donnez les adresses des variables `CH`, `D1`, `F1`, `D2` et `F2`. **(2,5 points)**

(h) Par quels déplacements par rapport au compteur de programme sont remplacés les étiquettes `ptrD1` et `ptrF1` en lignes 11 et 13 ? **(2 points)**

(i) Quelles valeurs sont affichées aux lignes 26 et 28 ? **(2 points)**

3.2 Schéma de Horner (5 points)

Nous rappelons que le code ASCII du caractère a se note 'a'. Le schéma de Horner permet de convertir une chaîne de caractères, par exemple la chaîne « 43434145 » représentant un entier en l'entier lui-même, ici 43434145.

Nous rappelons qu'en langage d'assemblage `ARM` (comme dans la plupart des langages) une chaîne de n caractères est représentée par un tableau de $n+1$ octets : les n premiers correspondent aux codes ASCII des caractères de la chaîne et le dernier est l'octet 0. Par exemple la chaîne "Bonjour" est stockée dans un tableau de 8 cases, contenant successivement les codes ASCII 'B', 'o', 'n', 'j', 'o', 'u', 'r' et enfin l'octet 0.

L'algorithme du schéma de Horner est donné ci-dessous. Ce code convertit la chaîne de caractères CH en un entier stocké dans r.

```
i:=0
r:=0
tant que ch[i] != 0 faire
     r=r*10+ch[i]-'0'
     i++
fin tant que
```

Nous rappelons que `mul r1, r2, r3` multiplie les contenus des registres `r2` et `r3` et range le résultat dans le registre `r1`. De plus en langage d'assemblage `ARM`, la valeur immédiate représentant le code ASCII du caractère '0' se note #'0'.

Questions :

(j) Complétez le code ci-dessous pour calculer et afficher la valeur entière représentée dans CH. Pour les variables i et r, vous utiliserez les registres $r2$ et $r3$, respectivement. **(4,5 points)**

(k) Proposer une variante dans laquelle la multiplication par 10 n'utilise ni instruction mult ni boucle **(0,5 point)**.

```
        .data
CH:     .asciz "43434145"
        .text
        .global main
main:
        [ Partie à compléter ]
        bal exit
PtrCH:  .word CH
```

ANNEXES

A Instructions du processeur ARM

Code	Nom	Explication du nom	Opération	Remarque
0000	`AND`	AND	et bit à bit	
0001	`EOR`	Exclusive OR	ou exclusif bit à bit	
0010	`SUB`	SUBstract	soustraction	
0011	`RSB`	Reverse SuBstract	soustraction inversée	
0100	`ADD`	ADDition	addition	
0101	`ADC`	ADdition with Carry	addition avec retenue	
0110	`SBC`	SuBstract with Carry	soustraction avec emprunt	
0111	`RSC`	Reverse Substract with Carry	soustraction inversée avec emprunt	
1000	`TST`	TeST	et bit à bit	pas `rd`
1001	`TEQ`	Test EQuivalence	ou exclusif bit à bit	pas `rd`
1010	`CMP`	CoMPare	soustraction	pas `rd`
1011	`CMN`	CoMpare Not	addition	pas `rd`
1100	`ORR`	OR	ou bit à bit	
1101	`MOV`	MOVe	copie	pas `rn`
1110	`BIC`	BIt Clear	et not bit à bit	
1111	`MVN`	MoVe Not	not (complément à 1)	pas `rn`
	`Bxx`	Branchement		xx = condition Cf. table ci-dessous
	`BL`	Branchement à un sous-programme		adresse de retour dans r14=LR
	`LDR`	"load"		
	`STR`	"store"		

L'opérande source d'une instruction `MOV` peut être une valeur immédiate notée `#5` ou un registre noté `Ri`, `i` désignant le numéro du registre. Il peut aussi être le contenu d'un registre sur lequel on applique un décalage de `k` bits ; on note `Ri, DEC #k`, avec `DEC` $\in$ {`LSL`, `LSR`, `ASR`, `ROR`}.

B Codes conditions du processeur ARM

La table suivante donne les codes de conditions arithmétiques `xx` pour l'instruction de branchement `Bxx`.

cond	mnémonique	signification	condition testée
0000	EQ	égal	Z
0001	NE	non égal	$\overline{Z}$
0010	CS/HS	$\geq$ dans N	C
0011	CC/LO	$<$ dans N	$\overline{C}$
0100	MI	moins	N
0101	PL	plus	$\overline{N}$
0110	VS	débordement	V
0111	VC	pas de débordement	$\overline{V}$
1000	HI	$>$ dans N	$C \wedge \overline{Z}$
1001	LS	$\leq$ dans N	$\overline{C} \vee Z$
1010	GE	$\geq$ dans Z	$(N \wedge V) \vee (\overline{N} \wedge \overline{V})$
1011	LT	$<$ dans Z	$(N \wedge \overline{V}) \vee (\overline{N} \wedge V)$
1100	GT	$>$ dans Z	$\overline{Z} \wedge ((N \wedge V) \vee (\overline{N} \wedge \overline{V}))$
1101	LE	$\leq$ dans Z	$Z \vee (N \wedge \overline{V}) \vee (\overline{N} \wedge V)$
1110	AL	toujours	

C Fonctions d'entrée/sortie

Nous rappelons les principales fonctions d'entrée/sortie du fichier `es.s`.

Les fonctions d'affichages :

— `bl EcrHexa32` affiche le contenu de `r1` en hexadécimal.
— `bl EcrZdecimal32` affiche le contenu de `r1` en décimal sous la forme d'un entier relatif de 32 bits.
— `bl EcrZdecimal16` affiche le contenu de `r1` en décimal sous la forme d'un entier relatif de 16 bits.
— `bl EcrZdecimal8` affiche le contenu de `r1` en décimal sous la forme d'un entier relatif de 8 bits.
— `bl EcrNdecimal32` affiche le contenu de `r1` en décimal sous la forme d'un entier naturel de 32 bits.
— `bl EcrNdecimal16` affiche le contenu de `r1` en décimal sous la forme d'un entier naturel de 16 bits.
— `bl EcrNdecimal8` affiche le contenu de `r1` en décimal sous la forme d'un entier naturel de 8 bits.
— `bl EcrChaine` affiche la chaîne de caractères dont l'adresse est dans `r1`.

Les fonctions de saisie clavier :

— `bl Lire32` récupère au clavier un entier 32 bits et le stocke à l'adresse contenue dans `r1`.
— `bl Lire16` récupère au clavier un entier 16 bits et le stocke à l'adresse contenue dans `r1`.
— `bl Lire8` récupère au clavier un entier 8 bits et le stocke à l'adresse contenue dans `r1`.
— `bl LireCar` récupère au clavier un caractère et stocke son code ASCII à l'adresse contenue dans `r1`.

D Table des codes ascii en décimal

32	␣	33	!	34	"	35	#	36	$	37	%	38	&	39	’
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	‘	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del