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

Contrôle continu UE INF241 : Introduction aux Architectures Logicielles et Matérielles

22 Mars 2011
Durée 1 h 30
Documents, calculettes, téléphones portables non autorisés
Le barême est donné à titre indicatif

1 Exercice : numération (5 points)

1. Donnez le code en binaire sur 8 bits des entiers relatifs suivants : `+100, +27, +50, -50, -79`
2. Pour chacune des additions suivantes : `(+100)+(+20)` et `(+100)+(+50)`,
   a) donnez le résultat apparent
   b) donnez les valeurs des indicateurs Z, N, C et V
   c) le résultat est-il correct ?
3. Pour chacune des soustractions suivantes : `(+27)-(+50)` et `(-79)-(+50)`,
   a) donnez le résultat apparent
   b) donnez les valeurs des indicateurs Z, N, C et V
   c) le résultat est-il correct ?

2 Exercice : codage en langage d'assemblage ARM (8 points)

On considère l'algorithme suivant qui calcule le nombre de 1 dans la représentation binaire d'un entier.

```
x, nb : entiers naturels   (* >= 0 *)

nb:=0
tantque x<>0 faire
   si x mod 2 <> 0 alors nb:=nb+1
   x:=x div 2
fin tantque
```

Question : Donnez le code de cet algorithme en langage d'assemblage ARM en implantant les variables `x` et `nb` dans la zone `data`. L'annexe II donne un résumé des instructions du processeur ARM.

3 Problème : mise en oeuvre d'un automate avec actions (7 points)

On considère l'automate avec actions reconnaisseur et évaluateur de nombres binaires à virgules étudié en cours. Les nombres sont écrits avec les caractères $\{0, 1, \bullet, \sqcup\}$. $\bullet$ est la virgule, $\sqcup$ est l'espace final. On se limite à des nombres avec au plus 4 chiffres avant la virgule et 4 après. On suppose qu'il n'y a pas d'erreurs.

On définit les variables suivantes :

— E est la valeur de **la partie entière naturelle du nombre** représenté.
— D est la valeur de **la partie naturelle après la virgule du nombre** représenté.
— N est **le nombre de chiffres après la virgule du nombre** représenté.

Par exemple pour 0110 • 1010␣, l'évaluation donne E=6 ; D=10 ; N=4.

Une description de cet automate est donnée en annexe III. Une table des codes ascii est donnée en annexe I.

On travaille avec un processeur relié à une mémoire organisée en octets dont les adresses sont sur 16 bits. Le processeur a 1 seul registre appelé accumulateur noté `Acc` de taille 8 bits. Il a un registre d'état qui comporte un bit : Z. La comparaison de l'accumulateur avec une valeur immédiate positionne le bit du mot d'état.

On convient que la lecture à l'adresse `0xC000` fournit un octet qui est le code ASCII du caractère courant. On range les valeurs `E`, `D` et `N` aux adresses `0x8000`, `0x8001` et `0x8002`.

Le jeu d'instruction ainsi que les tailles de codage sont décrits dans le tableau ci-dessous :

instruction	signification	codage
`load adr`	`Acc <-- Mem[adr]`	3 octets
`load# vi`	`Acc <-- vi`	2 octets
`store adr`	`Mem[adr] <-- Acc`	3 octets
`add# vi`	`Acc <-- Acc + vi`	2 octets
`lsl`	`Acc <-- Acc * 2`	1 octet
`cmp vi`	`Z<--1 si Acc-vi ==0; Z<--0 sinon`	2 octets
`beq depl`	si Z=1 `pc <-- pc + depl`	2 octets
`bne depl`	si Z=0 `pc <-- pc + depl`	2 octets
`jmp adr`	`pc <-- adr`	3 octets

Chaque séquence de code de gestion d'un état commence en mémoire à une adresse multiple de 100. Plus précisément la première instruction associée à l'état *Ini* est 100, la première instruction associée à l'état $E0$ est 200, puis respectivement 300, 400, 500 et 800 pour les états $E1$, $D0$, $D1$ et *Poi*.

Question : donnez le code pour cette machine correspondant à l'état `E1`.

Les étiquettes ne sont pas permises pour les branchements. Pour coder un saut après une comparaison avec un caractère utilisez un branchement relatif, pour coder un saut à un état utilisez un branchement absolu. Pour les branchements relatifs, notez que le compteur de programme (`pc`) au moment où le déplacement lui est ajouté repère l'instruction qui suit celle qui est en cours d'exécution.

4 ANNEXE I : table des codes ascii en hexadécimal

20	␣	21	!	22	”	23	#	24	$	25	%	26	&	27	’
28	(| 29 |)	2A	*	2B	+	2C	,	2D	-	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	_
60	‘	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	~	7F	del

5 ANNEXE II : instructions du processeur ARM

Nom	Explication du nom	Opération	remarque
`AND`	AND	et bit à bit	
`EOR`	Exclusive OR	ou exclusif bit à bit	
`SUB`	SUBstract	soustraction	
`RSB`	Reverse SuBstract	soustraction inversée	
`ADD`	ADDition	addition	
`ADC`	ADdition with Carry	addition avec retenue	
`SBC`	SuBstract with Carry	soustraction avec emprunt	
`RSC`	Reverse Substract with Carry	soustraction inversée avec emprunt	
`TST`	TeST	et bit à bit	pas `rd`
`TEQ`	Test EQuivalence	ou exclusif bit à bit	pas `rd`
`CMP`	CoMPare	soustraction	pas `rd`
`CMN`	CoMpare Not	addition	pas `rd`
`ORR`	OR	ou bit à bit	
`MOV`	MOVe	copie	pas `rn`
`BIC`	BIt Clear	et not bit à bit	
`MVN`	MoVe Not	not (complément à 1)	pas `rn`
`Bcc`	Branchement		cc = condition Cf. table ci-dessous
`BL`	Branchement à un sous-programme		adresse de retour dans r14=LR
`LDR`	"load"		
`STR`	"store"		

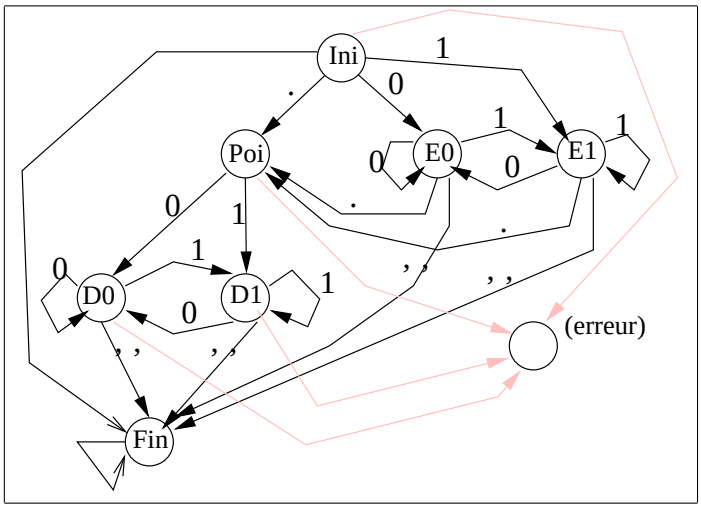
L'opérande source d'une instruction `MOV` peut être une valeur immédiate notée `#5` ou un registre noté `Ri`, `i` désignant le numéro du registre. Il peut aussi être le contenu d'un registre sur lequel on applique un décalage de `k` bits ; on note `Ri, DEC #k`, avec `DEC` $\in$ {`LSL`, `LSR`, `ASR`,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

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

mnémonique	signification	condition testée
EQ	égal	Z
NE	non égal	$\overline{Z}$
CS/HS	$\geq$ dans N	C
CC/LO	$<$ dans N	$\overline{C}$
MI	moins	N
PL	plus	$\overline{N}$
VS	débordement	V
VC	pas de débordement	$\overline{V}$
HI	$>$ dans N	$C \wedge \overline{Z}$
LS	$\leq$ dans N	$\overline{C} \vee Z$
GE	$\geq$ dans Z	$(N \wedge V) \vee (\overline{N} \wedge \overline{V})$
LT	$<$ dans Z	$(N \wedge \overline{V}) \vee (\overline{N} \wedge V)$
GT	$>$ dans Z	$\overline{Z} \wedge ((N \wedge V) \vee (\overline{N} \wedge \overline{V}))$
LE	$\leq$ dans Z	$Z \vee (N \wedge \overline{V}) \vee (\overline{N} \wedge V)$
AL	toujours	

6 ANNEXE III : automate reconnaisseur et évaluateur de nombres binaires

Les trois figures suivantes donnent respectivement une représentation graphique de l'automate, sa fonction de transition et sa fonction de sortie.



étatdedépart \ *symbole*	0	1	•	⊔
Ini	$E0$	$E1$	Poi	Fin
$E0$	$E0$	$E1$	Poi	Fin
$E1$	$E0$	$E1$	Poi	Fin
Poi	$D0$	$D1$		
$D0$	$D0$	$D1$		Fin
$D1$	$D0$	$D1$		Fin
Fin	Fin	Fin	Fin	Fin

Etat	action
Ini	$E \leftarrow 0$, $D \leftarrow 0$, $N \leftarrow 0$
$E0$	$E \leftarrow 2 \times E$
$E1$	$E \leftarrow 2 \times E + 1$
Poi	rien
$D0$	$D \leftarrow 2 \times D$; $N \leftarrow N + 1$
$D1$	$D \leftarrow 2 \times D + 1$; $N \leftarrow N + 1$
Fin	rendre la main