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

Contrôle continu UE INF241 : Introduction aux Architectures Logicielles et Matérielles

11 Mars 2013
Durée 1 h 30
Documents, calculettes, téléphones portables non autorisés
Le barême est donné à titre indicatif

1 Question de cours (2 points)

(a) Citez un exemple de périphérique d'entrée, de périphérique de sortie, de périphérique d'entrée/sortie. **(0,75 point)**
(b) Citez trois composants internes au processeur. **(0,75 point)**
(c) Quel est le nombre minimum de bits nécessaire pour adresser 600 mots mémoire de 1 octet dans une mémoire d'octets. **(0,5 point)**

2 Exercice : numération (4 points)

Pour cet exercice, **toutes les réponses devront être justifiées.**

(a) Donnez le code en binaire sur 8 bits des entiers relatifs suivants (codage en complément à deux) : $(-37)_{10}$ et $(+120)_{10}$. **(1 point)**
(b) Donnez le code en hexadecimal de $(56)_{10}$ et $(1011\ 1111\ 0010\ 0101)_2$. **(1 point)**
(c) Pour chacune des additions binaires suivantes : $(+120)_{10} + (-37)_{10}$ et $(+120)_{10} + (+37)_{10}$,
   1. donnez le résultat apparent sur 8 bits (codage en complément à deux), **(0,5 point)**
   2. donnez les valeurs des indicateurs Z, N, C et V. **(0,5 point)**
(d) Pour chacune des soustractions binaires suivantes : $(+120)_{10} - (+40)_{10}$ et $(37)_{10} - (37)_{10}$,
   1. donnez le résultat apparent sur 8 bits (codage en complément à deux), **(0,5 point)**
   2. donnez les valeurs des indicateurs Z, N, C et V. **(0,5 point)**

3 Exercice : codage en langage d'assemblage ARM (15 points)

Pour cet exercice, **toutes les réponses devront être justifiées** et **tous les codes écrits devront être commentés.**

(a) Quelle est la taille en nombre d'octets d'un tableau de 10 entiers relatifs codés sur 32 bits. **(0,25 point)**
(b) Donnez le code ARM permettant de déclarer un tableau *non initialisé* de 10 entiers relatifs codés sur 32 bits. **(1 point)**

(c) En supposant que le segment `bss` est stocké à l'adresse `20CF` et que le tableau précédent est stocké au début de ce segment, calculez l'adresse du troisième entier dans le tableau. **(1 point)**

(d) Donnez le code ARM permettant de remplir votre tableau avec des entiers relatifs saisis au clavier. **(2 points)**

(e) Donnez le code ARM permettant d'afficher les entiers relatifs contenus dans votre tableau. **(1 point)**

L'instruction ARM « `eor r1,r2,r3` » calcule le XOR bit-à-bit entre `r2` et `r3` et range le résultat dans **r1**. Ci-dessous, nous rappelons la table de vérité de l'opérateur XOR (ou exclusif).

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

(f) Donnez le code binaire en complément à deux de 63 sur 32 bits. **(0,5 point)**

(g) Donnez la valeur du mot binaire dans $r1$ après les instructions suivantes (pour l'instruction `mvn`, voir l'annexe) **(0,25 point)** :

```
mov r2,#63
mvn r3,#0
eor r1,r2,r3
```

(h) En utilisant l'instruction `eor`, donnez le code ARM qui remplace chaque entier dans le tableau par sa valeur absolue. **(1,5 point)**

À partir de maintenant, nous considérons que le tableau contient des entiers **naturels**.

(j) Donnez le code ARM calculant dans $r1$ et $r4$ l'indice et la valeur de l'élément le plus petit de votre tableau. **(2 points)**

(k) En supposant que $r1$ et $r2$ contiennent les adresses de deux entiers, donnez le code permettant d'échanger les deux entiers en mémoire. **(1,5 point)**

(l) En utilisant les codes précédents, donnez le code ARM pour trier votre tableau dans l'ordre croissant. **(4 points)**
L'idée est la suivante : cherchez l'indice i_{min} de l'élément le plus petit ($0 \leq i_{min} \leq 9$), puis échangez les valeurs situées aux indices 0 et i_{min}. Ainsi, la plus petite valeur se retrouve à l'indice 0. Recommencez en cherchant la valeur la plus petite entre les indices 1 et 9, puis faites l'échange, *etc.* En répétant 9 fois cette méthode, vous obtiendrez un tableau trié en ordre croissant.

4 ANNEXE I : 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`
`Bxx`	Branchement		xx = condition
			Cf. table ci-dessous
`BL`	Branchement à un		adresse de retour
	sous-programme		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 `xx` pour l'instruction de branchement `Bxx`.

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	

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.

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`.