

Contrôle Continu UE INF401 : Architectures des ordinateurs

Mars 2021, durée 1 h 30

Document : 1 A4 R/V personnel manuscrit autorisé ; calculatrices et téléphones portables interdits.
La plupart des questions sont indépendantes, si vous avez du mal avec l'une, passez à la suivante.
Tant que possible, indiquez bien tous les détails et justifiez vos réponses.
Le barème est donné à titre indicatif.

1 Numération et opération en binaire et en complément à 2 (6 points)

- Déterminer le nombre de bits minimum pour représenter ensemble les valeurs suivantes puis donner la représentation binaire de ces entiers relatifs avec le nombre de bits choisi globalement (codage en complément à 2) : $(+121)_{10}$, $(-323)_{10}$. **(2 points)**
- Donner les valeurs binaires et décimales des 2 entiers relatifs suivants codés sur 16 bits en complément à 2 : $(2021)_{16}$ et $(FA57)_{16}$. **(2 points)**
- Poser chacune des opérations suivantes sur 1 octet, effectuer l'opération en binaire (sur 1 octet, avec les retenues) et donner une interprétation de l'opération pour la représentation binaire usuelle et pour le codage en complément à 2 (sur 1 octet) ainsi que la valeur des indicateurs (Z, N, C et V) : $(0111\ 1011)_2 + (1110\ 1010)_2$ et $(0110\ 1111)_2 - (1101\ 1110)_2$ (soustraction par addition du complémentaire). Pour la soustraction, détailler la méthode employée. **(2 points)**

2 Programmation en ARM (8 points)

Le but de cet exercice est de calculer la $i^{\text{ème}}$ ligne du triangle de Pascal. Cette ligne sera stockée dans un tableau de **mots** représentant des entiers naturels. Ce tableau sera appelé **TAB**. L'algorithme vous permettant de résoudre ce problème est donné ci-dessous.

```
1. EcrChaine("Entrez un nombre'")
2. X:=Lire32()
3. si X <= 100 alors
4.   pour i de 0 à X-1 faire
5.     TAB[i] := 1
6.     j:=i
7.     tant que j >= 2 faire
8.       j := j - 1
9.       TAB[j] := TAB[j] + TAB[j-1]
10.    fin tant que
11.  fin pour
12. fin si
```

Dans l'algorithme, les variables `X`, `i` et `j` sont des entiers naturels sur 1 mot, les fonctions `EcrChaine` et `Lire32` correspondent aux fonctions du fichier `es.s` étudié en cours et en TP. Le fonctionnement des fonctions d'`es.s` est rappelé en annexe du sujet.

Vous allez maintenant traduire l'algorithme en assembleur ARM en vous basant sur le squelette de code donné ci-dessous.

```
.data
M: .asciz "Entrez un nombre"
.bss
X: .skip 4
TAB: .skip 400
.text
.global main
main:
    push {lr}

        @ partie manquante

    pop {lr}
    bx lr

ptr_X: .word X
ptr_M: .word M
ptr_TAB: .word TAB
```

Questions :

- Rappelez la taille d'un mot en octets. **(0,5 point)**
- Rappelez la différence entre les segments `.data` et `.bss` **(0,5 point)**
- Donnez le code ARM des instructions pour la ligne 1 (pour `EcrChaine`, vous appliquerez les conventions de `es.s` utilisées en TP notamment, *cf.* annexe). **(0,5 point)**
- En supposant que l'adresse de la variable `X` est stockée dans le registre `r1`, donnez le code ARM pour récupérer la valeur de la variable `X` dans le registre `r2` puis exécuter le test en ligne 3 (ne traduisez pas le bloc d'instruction des lignes 4-11). Si le test est faux, le branchement se fera vers l'étiquette `finCode`. **(1 point)**
- En supposant que l'adresse du tableau `TAB` est stockée dans le registre `r3`, que le registre `r0` représente la variable `i`, que le registre `r6` représente la variable `j`, donnez le code ARM des instructions des lignes 5 et 9. Utilisez les registres `r4`, `r5` et `r7` pour les calculs intermédiaires éventuels. **(2 points)**
- En supposant que l'adresse du tableau `TAB` est stockée dans le registre `r3`, que le registre `r0` représente la variable `i`, que le registre `r6` représente la variable `j`, donnez le code ARM des lignes 7-10 (hors ligne 9). Ne recopiez pas le code ARM de la ligne 9, inscrivez simplement à la place **@ ici ligne 9**. Introduire et utiliser une étiquette `apresTantQue` pour sortir du tant que. Utilisez les registres `r4`, `r5` et `r7` pour les calculs intermédiaires éventuels. **(2 points)**

- (g) En supposant que le registre `r0` représente la variable `i` et que la valeur de la variable `X` est stockée dans le registre `r2`, donnez le code ARM du « pour » en lignes 4-11. Ne recopiez pas le code ARM des lignes 5-10, inscrivez simplement à la place @ ici lignes (5-10). Introduire et utiliser une étiquette `apresPour` pour sortir du pour. (1.5 point)

3 Codage Négabinaire (6 points)

“Le système négabinaire (base -2) est un système de numération positionnel non standard utilisé dans l’ordinateur expérimental polonais BINEG, construit en 1957-59. Il possède la propriété inhérente d’avoir les nombres négatifs et positifs représentés sans un bit de signe” source Wikipédia.

Le négabinaire (base -2), c’est comme le binaire (base 2), les chiffres possibles sont aussi les bits 0 et 1. La différence ? en binaire, un nombre $(N_3N_2N_1N_0)_2$ a pour valeur $\sum_i N_i \times 2^i$; en négabinaire, un nombre $(N_3N_2N_1N_0)_{-2}$ a pour valeur $\sum_i N_i \times (-2)^i$.

Questions :

- (a) Donnez les valeurs décimales des 16 premiers nombres binaires et négabinaires sur 4 bits. Faites un tableau à trois colonnes (1 : combinaisons binaires sur 4 bits [dans l’ordre], 2 : valeurs associées en binaire habituel, 3 : valeurs associées en négabinaire). Encadrez les lignes où la valeur binaire est égale à la valeur négabinaire. (2 points)
- (b) Donnez le plus grand nombre sur 8 bits, en binaire, en hexadécimal et en décimal dont la valeur en binaire est égale à la valeur en négabinaire (1 points).
- (c) Dénombrez le nombre de demi-mots de 16 bits où la valeur binaire est égale à la valeur négabinaire. Expliquez votre calcul. (1 points).

Le négabinaire (base -2), c’est comme le binaire (base 2), pour trouver le codage d’un nombre il “suffit” d’utiliser le même algorithme de divisions successives : en binaire, des divisions successives par 2 ; en négabinaire, des divisions successives par -2. Attention, cependant, ne pas se tromper dans les divisions par -2 des nombres impairs (rappel, le reste ne peut prendre que les 2 valeurs possibles 0 et 1) : 7 divisé par -2 vaut -3 reste 1 ; -9 divisé par -2 vaut 5 reste -1.

Questions :

- (d) Appliquer l’algorithme des divisions successives par -2 pour 22 et donner la décomposition négabinaire de 22 sur 8 bits. (1 points)
- (e) Appliquer l’algorithme des divisions successives par -2 pour -34 et donner la décomposition négabinaire de -34 sur 8 bits. (1 points)

ANNEXES

A Fonctions d’entrée/sortie

Nous rappelons les principales fonctions d’affichagees du fichier `es.s` :

- `b1 EcrHexa32` affiche le contenu de `r1` en hexadécimal.
- `b1 EcrZdecimal32` (resp. `EcrZdecimal16`, `EcrZdecimal8`) affiche le contenu de `r1` en décimal sous la forme d’un entier relatif de 32 bits (resp. 16 bits, 8 bits).

- `bl EcrNdecimal32` (resp. `EcrNdecimal16`, `EcrNdecimal8`) affiche le contenu de `r1` en décimal sous la forme d'un entier naturel de 32 bits (resp. 16 bits, 8 bits).
- `bl EcrChaine` affiche la chaîne de caractères dont l'adresse est dans `r1`.

B Principales 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	
1000	TST	TeST	et bit à bit	pas rd
1001	TEQ	Test EQivalence	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	Branch	branchement conditionnel	xx = condition Cf. table ci-dessous
	LDR	LoaD Register	lecture mémoire	
	STR	STore Register	écriture mémoire	

L'opérande source d'une instruction MOV peut être une valeur immédiate notée #5 ou un registre noté R_i , 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 $R_i, DEC \#k$, avec $DEC \in \{LSL, LSR, ASR, ROR\}$.

C Codes conditions du processeur ARM

Nous rappelons 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	\bar{Z}
0010	CS/HS	\geq dans N	C
0011	CC/LO	$<$ dans N	\bar{C}
0100	MI	moins	N
0101	PL	plus	\bar{N}
0110	VS	débordement	V
0111	VC	pas de débordement	\bar{V}
1000	HI	$>$ dans N	$C \wedge \bar{Z}$
1001	LS	\leq dans N	$\bar{C} \vee Z$
1010	GE	\geq dans Z	$(N \wedge V) \vee (\bar{N} \wedge \bar{V})$
1011	LT	$<$ dans Z	$(N \wedge \bar{V}) \vee (\bar{N} \wedge V)$
1100	GT	$>$ dans Z	$\bar{Z} \wedge ((N \wedge V) \vee (\bar{N} \wedge \bar{V}))$
1101	LE	\leq dans Z	$Z \vee (N \wedge \bar{V}) \vee (\bar{N} \wedge V)$
1110	AL	toujours	