UNIVERSITÉ GRENOBLE ALPES
Licence de Sciences et Technologie - Parcours MIN, INM

Contrôle Continu UE INF401 : Architectures des ordinateurs

Mars 2022, durée 1 h 30
Document : 1 A4 R/V personnel manuscrit autorisé ; calculettes 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)

Rappel important : en machine, les entiers relatifs ($\mathbb{Z}$) sont représentés selon la méthode dite du complément à 2 (C2), il ne faut pas confondre cette méthode de représentation avec l'opération dite de complémentation.

(a) Déterminer le bon nombre de bits pour représenter toutes les valeurs suivantes puis donner la représentation binaire de ces entiers relatifs avec le nombre de bits choisi (codage en complément à 2) : $(+222)_{10}$, $(-201)_{10}$. **(2 points)**

Réponse : Il faut l'intervalle [-256,+255] donc 9 bits

$(+222)_{10} = (0\ 1101\ 1110)_2$

$(-201)_{10} = (1\ 0011\ 0111)_2$

(b) Donner la représentation binaire et la valeur décimale des 2 entiers relatifs suivants codés en complément à 2 sous forme hexadécimale : $(021E)_{16}$ et $(FAFA)_{16}$. **(2 points)**

Réponse : $(021E)_{16} = 542$
$(FAFA)_{16} = -1286$

(c) Effectuer, en donnant tous les détails (dont les retenues), les deux opérations suivantes, sur 1 octet : $(0011\ 1010)_2 + (0110\ 1110)_2$ et $(1110\ 1011)_2 - (0101\ 1011)_2$.
Pour chacune d'elle, donner une interprétation du résultat pour la représentation binaire usuelle et une interprétation pour le codage en complément à 2.
Pour chacune d'elles, donner la valeur des indicateurs (Z, N, C et V).
Pour la soustraction, effectuer l'opération par addition du complémentaire. **(2 points)**
Réponse :

L'addition

```
          0011 1010
        + 0110 1110
C=0 !=    1111 1100
V=1  ^    ---- ----
Z=0 N=1->1010 1000 =      :   0x   a8 --> 168 (IN) or -88 (ZZ)
```

La soustraction par addition du complémentaire

```
                                                    if natural     if
                                                       signed
          1110 1011
        + 1010 0100
   C=1 == 1101 1111 < c0=1   (in carries)
   V=0  ^ ---- ----
 Z=0 N=1->1001 0000  =     :  0x 90 --> 144 (IN) or -112 (ZZ)
```

La soustraction (non demandée) directe.

```
         1110 1011
       - 0101 1011
B=0 ==   0010 0000 < b0=0   (in borrows)
V=0  ^   ---- ----
Z=0 N=1->1001 0000
```

2 Rotations multiples d'un texte (8 points)

Cet exercice a pour objectif de programmer n rotations d'un texte. Exemple pour 3 rotations : "Bonjour le monde" => "onjour le mondeB", "njour le mondeBo", "jour le mondeBon". L'algorithme suivant est donc à traduire (le texte est dans la variable txt).

```
 1. EcrChaine(txt)
 2. ALaLigne
 3. si txt[0]==0 alors
 4.    EcrChaine(''Chaine vide'')
 5.    ALaLigne
 6. sinon si n==0 alors
 7.    EcrChaine(''Pas de rotation'')
 8.    ALaLigne
 9. sinon
10.    pour j= 1 .. n faire
11.       c := txt[0]
12.       i := 1
13.       tant que txt[i]!=0 faire
14.          t[i-1] := t[i]
15.          i := i+1
16.          fin tant que
17.       t[i-1] := c
18.       EcrNdecimal32(j)
19.       EcrChaine(''=>'')
20.       EcrChaine(txt)
21.       ALaLigne
```

```
22.       fin pour
23.    finsi
```

Dans l'algorithme, les variables `n, i` et `j` sont des entiers naturels sur 1 mot, les fonctions `EcrChaine` et `ALaLigne` 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 et l'allocation globale des registres suivante : i dans R5, j dans R6, c dans R7.

```
  .data
txt: .asciz "Bonjour le monde"
  .balign 4
n: .word 3

  .rodata
msgVide: .asciz "Chaine vide"
msg0: .asciz "Pas de rotation"
msgDonne: .asciz "=>"

  .text
  .global main
main:
  push {lr}

        @ partie manquante

  pop {lr}
  bx lr

LD_txt: .word txt
LD_n: .word n
LD_msgVide: .word msgVide
LD_msg0: .word msg0
LD_msgDonne: .word msgDonne
```

Questions :

(d) Quelle est la place mémoire occupée par le texte ? **(0,5 point)**.
Réponse :

> "Bonjour le monde" occupe 16 caractères puis `\0` donc 17 octets. Calé à 20 octets avec `.balign`

(e) Rappelez le sens de la directive `.balign`. Selon le cas, expliquer pourquoi la directive `.balign` est nécessaire ou au contraire pourquoi on pourrait l'omettre dans le cas présent. **(1 point)**.
Réponse :

`.balign` permet de recaler l'occupation mémoire sur des multiples d'octets. Nous pourrions nous en passer si la chaine de caractère avait un nombre de caractère multiple de 4 (à 1 près pour `\0`).

(f) Donner le code `ARM` pour les affichages (lignes 18, 19, 20, 21) **(1 point)**. **Réponse :**

```
1        mov r1, r6            @ copie de i dans r1
2        bl  EcrNdecimal32     @ affichage
3        ldr r1, LD_msgDonne   @ charge adresse de msgDonne
4        bl  EcrChaine         @ affichage
5        ldr r1, LD_txt        @ charge adresse du texte
6        bl  EcrChaine         @ affichage
7        bl  ALaLigne          @ affichage
```

(g) Donner le code `ARM` complet pour l'exécution du test de la ligne 3 ; si le test est vrai, le branchement sera vers l'étiquette `affichageTexteVide` sinon le branchement sera vers l'étiquette `testZeroRotation` **(1 point)**.
Réponse :

```
1      ldr  r0, LD_txt    @ r0 <- adresse de base du texte
2      ldrb  r7, [r0]      @ récupération du premier caractère
3      cmp  r7, #0
4      beq  affichageTexteVide
5      b  testZeroRotation
```

ou

```
1      ldr  r0, LD_txt    @ r0 <- adresse de base du texte
2      ldrb  r7, [r0]      @  récupération du premier caractère
3      cmp  r7, #0
4      bne  testZeroRotation
5      b  affichageTexteVide
```

ou

```
1      ldr  r0, LD_txt    @ r0 <- adresse de base du texte
2      ldrb  r7, [r0]      @ récupération du premier caractère
3      cmp  r7, #0
4      bne  testZeroRotation
5  affichageTexteVide:
6        ...
7      b fin
8  testZeroRotation:
9       ...
10 fin:
```

(h) Donner le code `ARM` complet du décalage de caractère dans le texte (ligne 14) **(1 point)**.

Réponse :

```
1      ldr  r0, LD_txt    @ r0 <- adresse de base du texte
2      add  r1, r0, r5    @ r1 <- adresse du caractère à déplacer
3      ldrb r2, [r1]      @ r2 <- le caractère à déplacer
4      sub  r3, r1, #1    @ r3 <- adresse où déplacer
5      strb r2, [r3]      @ stockage du caractère (c'est un octet)
```

(i) Donner le code `ARM` partiel de la gestion de la boucle "pour" (ligne 10 et 22 seulement), le corps de la boucle sera remplacé par un commentaire `@ici corps de la boucle` **(1.5 point)**.

Réponse :

```
1          ldr r4, LD_n
2          ldr r4, [r4]
3          mov r6, #1          @ init boucle
4  loop:   cmp r6, r4
5          bgt endloop
6
7          @ici corps de la boucle
8
9          add r6, r6, #1       @ j = j+1
10         b loop
11 endloop:@suite
```

(j) Donner le code `ARM` complet de la boucle "tant que" (ligne 13, 14, 15 et 16) **(2 points)**.

Réponse :

```
1          ldr  r0, LD_txt    @ r0 <- adresse début du texte
2  tq:     add  r1, r0, r5    @ r1 <- addresse du caractère
3          ldrb r2, [r1]        @ r2 <- le caractère
4          cmp  r2, #0        @ fin de la chaine
5          beq  fintq
6          sub  r3, r1, #1    @ nouvelle adresse
7          strb r2, [r3]      @ stockage du caractère avant
8          add  r5, r5, #1    @ indice suivant
9          b  tq
10 fintq:  @suite
```

3 Du binaire au ternaire (6 points)

«Le système ternaire (ou trinaire) est le système de numération utilisant la base 3. Les chiffres ternaires sont connus sous le nom de trit (trinary digit), de manière analogue à bit.» source Wikipédia.

Pour cet exercice, nous rappelons les premières puissances de 2 et de 3, faites-en bon usage.

n	0	1	2	3	4	5	6	7	8
2^n	1	2	4	8	16	32	64	128	256
3^n	1	3	9	27	81	243	729	2187	6561

3.1 Base 3

La base 3 usuelle est similaire à la base 2 pour représenter les entiers naturels ($\mathbb{N}$), avec les adaptations suivantes : les chiffres autorisés sont 0, 1 et 2 ; ce sont les puissances de 3 qui sont utilisées ; l'algorithme des divisions successives s'applique en effectuant des divisions par 3 ; etc.

Questions (2 points) :

(k) Donner la valeur décimale du nombre $(2022)_3$ représenté en base 3.
Réponse :

$(2022)_3 = 2 \times 3^3 + 0 \times 3^2 + 2 \times 3^1 + 2 \times 3^0 = 54 + 6 + 2 = 62$

(l) En base 3 combien de trits sont nécessaires pour écrire $(2022)_{10}$ représenté en base 10 ?
Réponse :

Il faut 7 trits car $(2022)_{10} < (2187)_{10} = 3^7$

(m) Écrire $(2022)_{10}$ en base 3.
Réponse :

```
2022 | 3
 22  |---
  12 | 674 | 3
   0 |  14 |---
         2 | 224 | 3
               14 |---
                2 | 74 | 3
                     14 |---
                      2 | 24 | 3
                            0 |---
                              | 8 | 3
                                2 |---
                                  | 2
```

$(2202220)_3 = 2 \times 3^6 + 2 \times 3^5 + 2 \times 3^3 + 2 \times 3^2 + 2 \times 3$
$(2202220)_3 = 2 \times 729 + 2 \times 243 + 2 \times 27 + 2 \times 9 + 2 \times 3$
$(2202220)_3 = 1458 + 486 + 54 + 18 + 6 = 2022$

(n) Combien de trits sont nécessaires pour écrire les nombres naturels ($\mathbb{N}$) représentés en binaire (base 2) sur 32 bits ?
Réponse :

Il faut 21 trits : $3^{20} = 3\ 486\ 784\ 401; 2^{32} = 4\ 294\ 967\ 296; 3^{21} = 10\ 460\ 353\ 203$
Sans calcul exact, $3^7 \simeq 2000$; donc $3^{21} \simeq 8\ 000\ 000\ 000 > 2^{32}(4Gi)$; L'approximation de $3^{21}/3$ est $< 4Gi$ donc trop petit.

3.2 Complément à 3 (C3)

Le complément à 3 est similaire au complément à 2, il est défini pour représenter les entiers relatifs ($\mathbb{Z}$). Il repose sur les mêmes principes : il y a autant d'entiers positifs que d'entiers négatifs ; les entiers positifs sont représentés en base 3 usuelle ; pour obtenir un entier négatif, il faut prendre le complément à 3^n. Ainsi $X + (-X) = 3^n$ et pour un codage en base 3 sur n trits, cela donne $X + (-X) = 0$ ((-X) est donc bien l'opposé de X).

Questions (2 points) :

(o) Donner la valeur décimale du nombre $(2022)_{C3}$ représenté en complément à 3 sur 4 trits.
Réponse :

Sur 4 trits, le C3 est un complément à $3^4 = 81$.
Pour avoir autant de nombres positifs que de nombres négatifs, comme en C2, les "grands" nombres sont négatifs.
C'est le cas de $(2022)_{C3}$, il faut donc prendre le complément de $(2022)_{C3}$ à 81 ; 81-62=19, donc la valeur recherchée est -19.
Autre solution (en passant par le ternaire) $X+(-X) = 3^n$ donc $(1000)_3-(2022)_{C3} = (-X)$

```
    1 0 0 0 0
-     2 0 2 2
    ---------
    1 1 1 1 0
    ---------
    0 0 2 0 1    => 2*3 + 1 = 19 => X = -19
```

Autre solution, en comptant sur les doigts (de Mickey !) et en partant du plus grand nombre avant 0 (donc de -1, comme en C2) :

```
2222 -> -1
2221 -> -2
2220 -> -3
2212 -> -4
2211 -> -5
2210 -> -6
2202 -> -7
2201 -> -8
2200 -> -9
2122 -> -10
2121 -> -11
2120 -> -12
2112 -> -13
2111 -> -14
2110 -> -15
2102 -> -16
2101 -> -17
2100 -> -18
2022 -> -19
```

(p) Donner le codage de l'opposé de $(2022)_{C3}$ représenté en complément à 3 sur 4 trits. (conseil : vérifier votre résultat en faisant l'addition $X + (-X)$)
Réponse :

En passant par les valeurs en décimal, on part de $(2022)_{C3} = (-19)_{10}$, il reste à code 19, on obtient $19 = 2 \times 3^2 + 1 = (0201)_3$.

Et effectivement :

```
  2 0 2 2
+ 0 2 0 1
  -------
  1 1 1 0
  -------
  0 0 0 0
```

Autre solution, sans passer par les valeurs en décimal, en inventant une opération similaire au complément à 2 : inverser les 2 et les 0 (les 1 restent tels que), et on ajoute 1. Ca marche aussi !

(q) Peut-on déterminer le signe d'un nombre représenté en complément à 3 sur 4 trits ? Si oui, comment ?
Réponse :

Comme il doit y avoir autant de nombres positifs que négatifs, le plus simple est de découper entre les premiers nombres 0000 ... 1111 et les suivants 1111 ... 2222 (à choisir, si 1111 est positif ou négatif). Le signe d'un nombre est donc le premier trit qui ne vaut pas 1 (en lisant en commençant par les poids forts).
Remarque, il y aurait aussi d'autres manières de découper en positifs et négatifs, dont

— un tiers et deux tiers (le chiffre de poids fort donne le signe) :
  (a) $x \leq (1222)_3$ pour être positif. Il est négatif sinon.
  (b) $x \leq (2222)_3$ pour être positif. Il est négatif sinon. C'est probablement la version la plus intéressante.

.

3.3 Notation ternaire équilibrée (3eq).

La notation ternaire équilibrée est une forme de numération de position issue de la base 3 qui utilise les chiffres -1, 0 et +1 au lieu de 0, 1 et 2. Ainsi la notation ternaire équilibrée permet de représenter les nombres négatifs. N. B. : -1 sera représenté par ⊖, 0 sera représenté par ⓪ et +1 sera représenté par ⊕.

Questions (2 points) :

(r) Donner la valeur décimale du nombre $(\oplus\ominus\oplus\oplus)_{3eq}$ représenté en notation ternaire équilibrée sur 4 trits équilibrés.
Réponse :

$(\oplus\ominus\oplus\oplus)_{3eq} = 1 \times 3^3 + (-1) \times 3^2 + 1 \times 3 + 1 = 27 - 9 + 3 + 1 = 22$

(s) Donner, si c'est possible, le codage de $(-1)_{10}$ en notation ternaire équilibrée sur 4 trits équilibrés.
Réponse :

$(\ominus\ominus\ominus\ominus)_{3eq} = -1$

(t) Quel est l'intervalle des nombres relatifs représentés par la notation ternaire équilibrée sur 4 trits équilibrés ?
Réponse :

max : $(\oplus\oplus\oplus\oplus)_{3eq} = 1\times3^3+1\times3^2+1\times3^1+1\times3^0 = 27+9+3+1 = 40$
min : $(\ominus\ominus\ominus\ominus)_{3eq} = -1\times3^3+(-1)\times3^2+(-1)\times3^1+(-1)\times3^0 = -27-9-3-1 = -40$

(u) Peut-on déterminer le signe d'un nombre écrit en notation ternaire équilibrée sur 4 trits équilibrés ? Si oui, comment ?
Réponse :

Plusieurs réponses possibles :

- On peut le convertir en décimal et comparer à 0...
- C'est le signe du trit de poids le plus fort non nul. (les trits de poids inférieurs ne peuvent compenser).
- Il y a un lien (dans les deux sens) entre 3eq et C3, on peut aussi se ramener à la question précédente en C3 (ou inversement, on peut répondre à la question de C3 en fonction de la réponse ici)

4 Commentaires libres

- **Notes.** moyenne des 50 premières copies : 10 (notes entre 2 et 19)
- **Erreurs (habituelles).**
  - Confusion entre la représentation C2 et l'opération C2 (15% des copies, malgré l'indication en début de sujet).
  - Confusion entre mov (entre registres) et ldr (entre registre et mémoire)
  - Confusion entre ldr (lecture à partir de la mémoire) et str (écriture en mémoire)
  - Confusion entre b et bl
  - Erreur de taille, ldr vs ldrb, selon entier ou adresse = 32 bits = 4 octets = 1 mot et caractère = 1 octet = 8 bits
  - Chaînes : les espaces comptent, à la fin un caractère NULL est ajouté pour utiliser les fonctions d'affichages standards
  - Oubli des sauts inconditionnels en fin de boucle et entre "alors" et "sinon"
  - Oubli de l'initialisation et/ou de l'incrémentation du compteur dans les boucles "pour"
- **Autres (erreurs).**
  - La première question donne des réponses de taille variée : entre 2 lignes et plus d'une page (50 lignes)

- Attention pour les conditions qui ne sont pas immédiatement liées à une comparaison : mauvais placement pour les étiquettes de début de boucle (sur le début de lecture txt[i] et non pas sur le cmp). Souvent, les boucles ont une condition simple et l'étiquette est sur le cmp, mais ce n'est pas toujours le cas.
- La vérification de la validité du C3 (opération) n'a pas été souvent effectuée (ou montrée) et paradoxalement elle n'a pas eu l'effet escompté, sur 5 confirmations, 2 seulement sont correctes (biais de confirmation ?)
- Confusion entre C3 et 3eq (comme si 3eq n'était qu'une représentation de C3)
- Orthographe : bit prend un s au pluriel : 8 bits (et pas 8 bit).

— **Conseils.**

- Bien lire l'énoncé
- Appliquer les motifs de traductions
- Garder son bon-sens
- Garder un œil critique
- Appliquer des méthodes de contrôle et de vérification

ANNEXES

A Fonctions d'entrée/sortie

Nous rappelons les principales fonctions d'affichages du fichier `es.s` :

- `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`.
- `bl ALaLigne` provoque un passage à la ligne dans l'affichage.

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 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`	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 `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`}.

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	$\overline{Z}$
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	