1- Représentation et arithmétique

Objectifs

L'objectif est de se familiariser avec les types de données simples, les opérations et instructions de traitement de base. Pour certains exercices, les rappels fournis en annexe pourront vous être utiles.

Un processeur ARM est capable de faire des additions, soustractions et opérations logiques sur des nombres entiers de taille 32 bits en complément à deux. Toutes ces opérations sont effectuées par l'Unité Arithmétique et Logique (UAL, en anglais ALU). Une description des instructions assembleur réalisant ces opérations arithmétiques et logiques vous est fournie en annexe.

Il n'est pas possible d'effectuer directement des opérations sur des données rangées en mémoire. Pour une opération arithmétique (ou logique), un des deux opérandes est toujours pris dans un registre du processeur, l'autre opérande peut être soit une valeur immédiate, soit la valeur d'un registre, soit la valeur d'un registre ayant subi un décalage (ou une rotation). Le résultat est rangé dans un registre. La forme générale d'une instruction est donc la suivante :

```
INST Rd, Rn, <op2>
```

`Rd`: registre de destination

`Rn`: registre contenant la valeur du premier opérande

`<op2>` : représente le deuxième opérande qui peut être :

- `#literal` : une valeur immédiate (par exemple #45). Ex : `ADD R3, R2, #45`
- `Rm` : le nom d'un registre qui contient la valeur de l'opérande. Ex : `ADD R3, R2, R4`
- `Rm, shift` : un registre associé à une opération de décalage. Ex : `ADD R3, R2, R5, LSL #2`

1.1 Représentation numérique

Donner la représentation de la donnée 'FEED'

a) En binaire (interprété comme un nombre hexadécimal)

b) En décimal

c) En binaire (interprété comme code ASCII)

Donner la représentation du nombre binaire 11011110

a) En décimal

b) En hexadécimal

c) Quel est le résultat (en binaire et en hexadécimal) :

- d'un décalage logique à droite (LSR) de deux positions (sur 8 bits)
- d'un décalage arithmétique à droite (ASR) de deux positions (sur 8 bits)
- d'un décalage logique à gauche (LSL) de deux positions
- d'un décalage arithmétique à gauche de deux positions

Donner la représentation du nombre décimal 996 :

a) En binaire

b) En hexadécimal

c) Quel est le résultat (en binaire et en hexadécimal)

- d'un décalage logique à droite (LSR) de deux positions (sur 16 bits)
- d'un décalage arithmétique à droite (ASR) de deux positions (sur 16 bits)
- d'un décalage logique à gauche (LSL) de trois positions
- d'une rotation à droite (ROR) de trois positions (sur 16 bits)

1.2 Opérations arithmétiques et registre d'état

Donnez les valeurs des bits N, Z, C et V, dans le cas d'une addition et d'une soustraction, pour les valeurs de a et b suivantes :

a	b	a+b	N	Z	C	V
0x08000000	0x07000000					
0x08000000	0x08000000					
0x40000000	0x30000000					
0x40000000	0x40000000					
0x08000000	0xFFFFFFFF					
0xF0000000	0xFFFFFFFF					
0x80000000	0xFFFFFFFF					
0x80000000	0x80000000					

a	b	a-b	N	Z	C	V
0x08000000	0x04000000					
0x04000000	0x08000000					
0x08000000	0x08000000					
0xF0000000	0x7F000000					
0x7F000000	0xF0000000					

1.3 Instructions arithmétiques

On considère que les registres `R0`, `R1`, `R2`, `R3`, `R4`, `R5` du processeur contiennent les nombres hexadécimaux suivants :

```
R0      A000 0000
R1      F8FF EFFF
R2      8FFF 0000
R3      0000 000C
R4      FFFF FFFF
R5      7000 FFFF
```

Donner le contenu des registres `R8`, `R7`, `R6`, `R9`, `R1` (sous forme de nombres hexadécimaux) après exécution *successive* des instructions suivantes :

```
a) ADD      R8, R2, R5
b) ADD      R7, R4, R2, LSR #4
c) ADD      R6, R4, R2, ASR #4
d) SUBS     R9, R1, R6
e) SUBEQ    R1, R4, R3
```

Le suffixe `S` (dans l'instruction `SUBS`) indique que l'on met à jour les indicateurs `N`, `Z`, `C`, `V`.

Le suffixe `EQ` (dans l'instruction `SUBEQ`) indique que l'on exécute l'instruction si la condition `EQ` (EQual) est vérifiée. Cette condition correspond à l'indicateur `Z=1`.

1.4 Instructions logiques

Une opération de masquage consiste à forcer à 0 ou à 1 certains bits d'un mot tout en gardant les autres bits intacts. L'utilisation de masquages est très fréquente en programmation bas niveau. Elle permet notamment de forcer ou de tester la valeur de certains bits dans les registres des périphériques associés au processeur.

Dans les trois exercices suivants, vous devez décrire ce que fait le code suivant, instruction par instruction, puis dans sa globalité.

a)

```
MOV     R0, #0x3C
AND     R1, R0, #8
MOV     R0, #0x34
AND     R1, R0, #8
```

b)

```
MOV     R0, #0x45F702A8
AND     R1, R0, #0x000000FF
AND     R2, R0, #0x0000FF00
AND     R3, R0, #0x00FF0000
AND     R4, R0, #0xFF000000
```

c)

```
MOV     R0, #0x4C
MVN     R1, #0x08
AND     R0, r0, r1
```

1.5 Addition d'entiers longs

On cherche à effectuer l'addition de deux nombres entiers de 64 bits contenus respectivement dans les paires de registres 32 bits `(R1, R0)` et `(R3, R2)`. Le résultat sera placé dans `(R5, R4)`. On considère que les registres `R1`, `R3` et `R5` contiennent les 32 bits de poids fort de ces 3 nombres.

a) Ecrire une séquence assembleur réalisant une addition sur 64 bits.

b) Comment écrirait-on l'addition sur 128 bits ?

ANNEXE

Correspondance Décimal ASCII

Décimal	Caractère	Décimal	Caractère	Décimal	Caractère
32	ESPACE	48	0	91	[
33	!	49	1	92	\
34	"	…	…	63	]
35	#	57	9	…	
36	$	…	…	…	
37	%	65	A	97	a
38	&	66	B	98	b
39	'	…	…	…	…
40	(	90	Z	122	z

Registre d'état

C (Carry): indicateur positionné à 1 par l'UAL si l'opération a générée une retenue.

Z(Zero): indicateur positionné à 1 par l'UAL si le résultat renvoyé est nul.

N (Negative): indicateur positionné à 1 par l'UAL si le résultat renvoyé est négatif.

V (oVerflow): indicateur positionné à 1 par l'UAL en cas de dépassement de capacité de l'UAL.

Instructions arithmétiques et logiques (ARM)

Instruction	***Opération***	***Description***
`ADD Rd, Rn, <op2>`	`Rd ← Rn + <op2>`	"ADD" : addition
`ADC Rd, Rn, <op2>`	`Rd ← Rn + <op2> + (C)`	"ADD/Carry": addition+retenue
`SUB Rd, Rn, <op2>`	`Rd ← Rn – <op2>`	"SUBtract" : soustraction
`RSB Rd, Rn, <op2>`	`Rd ← <op2> – Rn`	"Reverse SuBtract" : soustraction renversée
`MOV Rd, <op2>`	`Rd ← <op2>`	«MOVe» : rangement d'une valeur dans un registre
`MVN Rd, <op2>`	`Rd ← non <op2>`	«MoVe Not» : négation logique bit à bit
`AND Rd, Rn, <op2>`	`Rd ← Rn et <op2>`	«AND» : *et* logique bit à bit
`ORR Rd, Rn, <op2>`	`Rd ← Rn ou <op2>`	«OR Register» : *ou* logique bit à bit
`EOR Rd, Rn, <op2>`	`Rd ← Rn oux <op2>`	«Exclusive OR» : *ou exclusif* bit à bit
`BIC Rd, Rn, <op2>`	**`Rd`** ← **`Rn`** et non **`<op2>`**	«BIt Clear» : masquage inverse

`<op2>` représente le deuxième opérande qui peut être:

- `#literal` : une valeur immédiate
- `Rm` : le nom d'un registre qui contient la valeur de l'opérande.
- `Rm, shift` : un registre associé à une opération de décalage.