Université Paris Sud D. Etiemble
Licence d'Informatique

ARCHITECTURE DES ORDINATEURS
PARTIEL Octobre 2013 (CORRIGE)
Tous documents autorisés – Calculettes autorisées.

PARTIE 1 : JEU D'INSTRUCTIONS MIPS32

Dans cette partie, on utilise le jeu d'instructions MIPS32.

On suppose que les registres R0 à R5 ont les contenus suivants, exprimés en hexadécimal :

R0	0000 0000
R1	8888 AAAA
R2	FEDC 3210
R3	0000 000C
R4	A800 0000
R5	C000 0000

Q 1) Donner les valeurs des registres modifiés après exécution des instructions suivantes.

a) ADD R6, R1,R2 R6 =8764DCBA
b) ADDU R7, R4, R5 R7 = 68000000 (Débordement : négatif + négatif donne positif)
c) ADD R8, R4, R5 Trappe débordement (Débordement : négatif + négatif donne positif)
d) SUB R8, R5, R3 R8 = BFFFFF4
e) SRA R9,R5,8 R9 = FFC0 0000
f) SRL R10,R2,16 R10 = 0000 FEDC

Q 2) Ecrire l'instruction ou la suite d'instructions qui place la constante 0x7FFFFFFF dans le registre R3

```
LUI R3 ,0x7FFF      //R3 reçoit 0x7FFF 0000
ORI R3,R3, 0xFFFF   //Ou logique entre 0x7FFF 0000 et 0x0000 FFFF
```

Q 3) On considère le programme suivant

```
        ADDI R1, R0, 0
        BGTZ R2, suite
        ADDI R1, R1,-1
Suite : ADD R1, R1, 1
```

Que contient le registre R1 après exécution du programme si R2 contient initialement

a) FFFFFFFF
b) 00000001

a) : 0000 0001 b) 0000 0000

PARTIE 2 : JEU D'INSTRUCTIONS ARM

Dans cette partie, on utilise les instructions ARM décrites en annexe.

Q 4) Donner l'instruction ou la suite d'instructions pour multiplier le contenu du registre R0 par

a) la constante 63 : RSB R0, R0, LSL #6
b) la constante 17 : ADD R0, R0, R0 LSL #4

Q5) Donner les instructions ARM pour effectuer

- Un décalage arithmétique à droite de R1 de 4 positions MOV R1,R1, ASR#4
- Un décalage logique à droite de R1 de 2 positions MOV R1,R1, LSR #2
- Un décalage logique à gauche de R1 de 4 positions MOV R1,R1,LSL #4

Soit une zone mémoire à partir de l'adresse C000 0000

Adresse (hexadécimal)	Contenu mot 32 bits (hexadécimal)
C000 0000	0x 12 34 56 78
C0000 0004	0x FE DC BA 98
C000 0008	0x 00 11 22 33
C000 000C	0x A1 B2 C3 D4
C000 0010	0x F9 E8 D7 C6

Q 6) Donner le contenu des registres ou des cases mémoire modifiées (MEM8, MEM16 ou MEM32) après exécution des instructions suivantes. On suppose l'ordre « little endian ». Le registre R1 contient 0x33335555, le registre R3 contient 0x00000008 et le registre R5 contient 0xC00000000 .

NB : les instructions s'exécute les unes après les autres.

NB : l'ordre « little endian » range le mot 0x0A0B0C0D dans l'ordre suivant dans les 4 premiers octets de la mémoire :

Octet d'adresse 3 : 0A
Octet d'adresse 2 : 0B
Octet d'adresse 1 : 0C
Octet d'adresse 0 : 0D

L'octet d'adresse 0 est donc 0D
Le mot de 16 bits d'adresse 0 est donc 0C0D
Le mot de 32 bits d'adresse 0 est donc 0A0B0C0D

a)	LDRSB R11, [R5 ,#4]	R11 = MEM8(C000 0004) = 0x FFFFFFFF98	R5 = C000 0000
b)	LDRH R12, [R5,R3]	R12 = MEM16 (C000 0008) = 0x 00002233	R5 = C000 0000
c)	LDR R13, [R5, #12]	R13 = MEM32 (C000 000C) = 0xA1B2C3D4	R5=C000 0000
d)	LDR R14, [R5], #4	R14= MEM32(C000 0000) = 0x12345678	R5=C000 0004
e)	LDR R6, [R5, #8] !	R6 = MEM32(C000 000C) = 0xA1B2C3D4	R5=C000 000C
f)	STR R1,[R5], #8	MEM(C000 000C) = 0X33335555	R5=C000 0014
g)	STRB R3, [R5,-R3]	MEM8 (C000 000C) = 0x0000 0008	R5=C000 0014

h) LDR R7, [R5, #6] ! Accès non aligné

Q 7**) Soit le programme ARM qui effectue un traitement sur deux tableaux d'entiers X et Y. Donner le programme C correspondant au programme assembleur ARM**
Quel est le résultat de l'exécution du programme.

```
      LDR r1,=X
      LDR r2,=Y
      MOV r3, #10
   LOOP: LDR r4,[r1], #4
      LDR r5,[r2],#4
      LDR r6,[r1],#4
      LDR r7,[r2],#4
      ADD r0,r5,r4
      STR r0,[r2, #-8]
      ADD r0,r7,r6
      STR r0,[r2,#-4]
      SUBS r3,r3,#2
      BGT LOOP
      SWI 0x11 @ Stop program execution

   .data
   X: .word 1,-3,5,-7,9,-11,13,-15,17,-19
   Y: .word 1, 2,3, 4,5,  6, 7,  8, 9, 10
```

```
For (i=0; i<10; i+=2){
      Y[i]=X[i] + Y[i];
      Y[i+1]=X[i+1] + Y[i+1];}
Y[10] = 2, -1, 8, -3, 14, -5, 20, -7, 26, -9
```

Q 8) Ecrire un programme ARM qui compte le nombre d'entiers strictement positifs dans un tableau de 10 entiers X. Le résultat sera retourné dans le registre R0.

```
      LDR r1,=X
      MOV r3, #10
LOOP: LDR r2,[r1], #4
      CMP R2, #0
      ADDGT r0, r0, #1
      SUBS r3,r3,#1
      BGT LOOP
      SWI 0x11 @ Stop program execution
```

Q 9) Ecrire un programme ARM qui compte le nombre de bits à 1 dans le registre R1 et met le résultat dans R0.

```
      MOV R0,#0
Boucle: AND R2,R1,#1    @Test du bit de poids faible de R1
      ADD R0,R0,R2
      MOV R1,R1,LSR#1   @ Décalage logique d'une position à droite
      CMP R1,#0
      BGT Boucle        @ Arrêt lorsqu'il n'y a plus de 1 dans R1

      SWI 0x11 @ Stop program execution
```

```
Autre solution
      MOV R0,#0
Boucle : CMP R1,#0
      ADDLT R0,R0,#1    @Le bit de signe est à 1
      MOV R1,R1,LSL#1   @décalage à gauche
      BNE Boucle        @ tant qu'il reste des 1 dans R1
      SWI 0x11          @ Stop program execution
```

Annexe : Jeu d'instructions ARMR1

On rappelle que le processeur ARM a 15 registres de 32 bits. Les immédiats sont signés.
R15 est le compteur de programme.

Instruction	Assembleur	Effet
MOV	MOV Ri,Rj	Ri ← Rj

ADD	ADD Ri, Rj, Rk
	ADD Ri, Rj, #N
	ADD Ri, Rj, Rk Décalage #N
		Ri ← Rj+Rk
		Ri ← Rj+N
		Ri ← Rj+(Rk décalé de N positions)

SUB	SUB Ri, Rj, Rk
	SUB Ri, Rj, #N
	SUB Ri, Rj, Rk Décalage #N
		Ri ← Rj - Rk
		Ri ← Rj - N
		Ri ← Rj – (Rk décalé de N positions)

RSB	RSB Ri, Rj, Rk RSB Ri, Rj, #N RSB Ri, Rj, Rk Décalage #N	Ri ← Rk - Rj Ri ← N - Rj Ri ← (Rk décalé de N positions) -Rj
AND (et logique)	AND Ri, Rj, Rk AND Ri, Rj, #N AND Ri, Rj, Rk Décalage #N	Ri ← Rj and Rk Ri ← Rj and N Ri ← Rj and (Rk décalé de N positions)

SUBS	SUBS Ri, Rj, Rk
	SUBS Ri, Rj, #N
	SUBS Ri, Rj, Rk Décalage #N
		Ri ← Rj - Rk et Rcc positionné
		Ri ← Rj - N et Rcc positionné
		Ri ← Rj – (Rk décalé de N positions) et Rcc positionné

ORR (ou logique)	ORR Ri, Rj, Rk ORR Ri, Rj, #N ORR Ri, Rj, Rk Décalage #N	Ri ← Rj or Rk Ri ← Rj or N Ri ← Rj or (Rk décalé de N positions)
EOR (ou exclusif)	EOR Ri, Rj, Rk EOR Ri, Rj, #N EOR Ri, Rj, Rk Décalage #N	Ri ← Rj xor Rk Ri ← Rj xor N Ri ← Rj xor (Rk décalé de N positions)
CMP	CMP Ri,Rj	Compare Ri et Rj (ou Ri et #N) et

	CMP Ri, #N	positionne le registre code condition
BGT	BGT adresse cible	Branchement si le résultat de la comparaison (CMP) était > ou si le résultat de SUBS était différent de 0.

Table 1 : Opérations arithmétiques et logiques utilisées

Les instructions mémoire utilisées sont données dans la table 2. L'adresse mémoire est donnée par le mode d'adressage indiqué dans la table 3. Mem32 signifie un accès mémoire à un mot de 32 bits. Mem16 signifie un accès mémoire à un demi-mot (16 bits). Mem8 signifie un accès octet. Dans le cas d'un accès Mem16 et Mem8, il est précisé si le registre de 32 bits est complété à gauche par des 0 (extension zéro) ou par le signe du demi-mot ou de l'octet lu (extension signe).

Instruction	Effet	Commentaire
LDR	Rn ← Mem32 (Adresse)	Chargement mot
LDRH	Rn ← extension zéro, Mem16 (Adresse)	Chargement demi mot non signé
LDRSH	Rn ← extension signe, Mem16 (Adresse)	Chargement demi mot signé
LDRB	Rn ← extension zéro, Mem8 (Adresse)	Chargement octet non signé
LDRSB	Rn ← extension signe, Mem8 (Adresse)	Chargement octet signé
STR	Mem32 (Adresse) ← Rn	Rangement mot
STRH	Mem16 (Adresse) ← Rn[15:0]	Rangement demi mot
STRB	Mem8 (Adresse) ← Rn[7:0]	Rangement octet

Table 2 : Instructions mémoire

Mode	Assembleur	Action
Déplacement 12 bits, Pré-indexé	[Rn, #deplacement]	Adresse = Rn + déplacement
Déplacement 12 bits, Pré-indexé avec mise à jour	[Rn, #deplacement] !	Adresse = Rn + déplacement Rn ← Adresse
Déplacement 12 bits, Post-indexé	[Rn], #deplacement	Adresse = Rn Rn ← Rn + déplacement
Déplacement dans Rm Préindexé	[Rn, ± Rm, décalage]	Adresse = Rn + décalage (Rm)
Déplacement dans Rm Préindexé avec mise à jour	[Rn, ± Rm, décalage] !	Adresse = Rn + décalage (Rm) Rn ← Adresse
Déplacement dans Rm Postindexé	[Rn], ± Rm, décalage	Adresse = Rn Rn ← Rn + décalage (Rm)

Table 3 : Modes d'adressage.

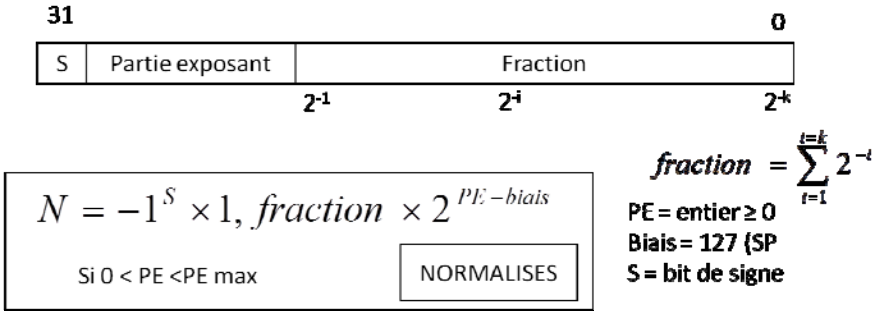

Figure 1 : Rappel sur le format flottant simple précision.