Systèmes Embarqués 1 : Travail écrit no 1. 4.9

Nom : **Prénom :** **Classe : T-2/I-2** **Date : 20.11.2013**

Problème n° 1 (Programmation en assembleur)

Codez en langage assembleur ARM l'algorithme ci-dessous (version droite ou gauche). Le code contiendra toutes les directives permettant un assemblage et un linkage correcte du module et de sa fonction.

```
char string[20];
uint32_t val = 0123456;

void o2a() {
    char* c = &string[19];
    do {
        *c-- = (val % 8) + '0';
        val /= 8;
    } while (val != 0);

    char* s = string;
    do {
        *s++ = *++c;
    } while (c < &string[19]);
    *s = 0;
}
```

--- en assembleur --------

```
char string[20];
uint32_t val = 0123456;

void o2a() {
    int i = 19;
    do {
        string[i--] = (val % 8) + '0';
        val /= 8;
    } while (val != 0);

    int j = 0;
    do {
        string[j++] = string[++i];
    } while (i < 19);
    string[j] = 0;
}
```

--- en assembleur --------

```
.global o2a
.data
.ascii result .space 20
.long val 0123456
.text
o2a: ldr r0, =val      @ int value      ldr r0, [r0]
     ldr r1, =result   @ resultat
     @ r2 -> char c
     @ r3 -> chars
     mov r4, #19
loop1: mov r2, r0, lsr #3
       mov r2, r2, lsl #4
       strb r2, [r1, r4]
       subs r4, r4, #-1
       eor r0, 0x7   mov r0, r0, lsr #3
       cmp r1, #0
       bne loop1
mov r3, #0          @ s=string
loop2: ldrb mov r3, [r2, #1]!   @ s = ++c
       mov r3, r3, #1           @ s++
       cmp r2, [r1, r4]
       blt loop2
       mov r3, #0
```

@ @ → (val % 8) + 0

@ c=

@ c--

@ val /= 8;

@ @ → while condition

[Gac/11.2014] T-2/I-2

Systèmes Embarqués 1 : Travail écrit no 1.

8

Problème n° 2 (Mode d'adressage)

a) Donnez les 2 instructions assembleur permettant de stocker le contenu des registres R3, R4, R5, R8 et R9 dans la structure « regs » ci-dessous

```
struct Regs {uint32_t r3, r4, r5, r8, r9} regs;
```

1,5 ldr r0, = regs
stmia regs!, {r3-r5, r8, r9}
r0

b) Donnez l'instruction assembleur permettant de restaurer l'état de la pile et retourner au programme appelant, sachant que les registres suivants ont été stockés sur la pile, soit : r4, r5 et lr (les instructions push et pop ne peuvent pas être utilisées) et selon les conventions utilisées durant les TPs.

1,5 ldmdb regs!, {r3-r5, r8, r9}

c) Pour le code assembleur et la représentation de la mémoire (**Big-Endian / 8-bits**) et l'état des registres du processeur ci-dessous, donnez le résultat des opérations (état des registres, état de la mémoire):

Mémoire (big-endian / 8 bits)		(après)
0xb0002100	0x34	0x01 ✓
0xb0002101	0xf5	0x00 ✓
0xb0002102	0x89	
0xb0002103	0xc9	
0xb0002104	0x25	
0xb0002105	0x94	
0xb0002106	0xa5	
0xb0002107	0xc2	
0xb0002108	0xba	
0xb0002109	0x53	
0xb000210a	0x41	
0xb000210b	0x87	

	Registres (avant)	Registres (après)
R0	0x0000'0100	
R1	0x0038'3004	
R2	0x0000'000c	0x0000'0008
R3	0x0000'12f8	0xffff'ffc2 ✓
R4	0x0000'0006	
R5	0xb000'2106	0xb000'2100 ✓
R6	0xb000'2107	0xb000'210a ✓
R7	0xa000'5101	
R8	0xa000'3008	
R9	0x0302'0100	
R10	0x0403'0200	
R11	0x0504'0300	
R12	0xa000'2008	
SP	0xa000'5110	

le reste ne change pas !

1. `sub r2,r1,r0,lsr #6` ✓

lsr(r0 = ...0000 0000 0000 0100) → 0x0000'0004 ✓

$r_2 = 0x0000'000c - 0x0000'0004 = 0x0000'0008$ (r1)

1,5

c = 12 − 4 = 8

2. `strh r0, [r5, -r4]!`

r5 = 0xb000'2100

2 met 0x0100 dans

3. `ldrsb r3, [r6],#0x3`

2 r3 = 0xffff'ffc2 r6 = 0xb000'210a

Systèmes Embarqués 1 : Travail écrit no 1.

128 64 32 16 8 4 2 1 — 86 112 120

9

Problème n° 3 (traitement numérique des nombres)

a) Prévoyez l'état des flags Z, C, N et V ainsi que le résultat contenu dans le registre R2 (en décimal) suite à l'exécution des instructions assembleur suivantes :

Remarque : toutes les opérations sont faites avec des registres de 8 bits au lieu de 32 bits

1.
```
ldr   r2, =104
adds  r2, #24
```

104 → 0110 1000
24 → 0001 1000
1000 0000

2

Z= 0 C= 0 N=1 V=1 R2(non signé)= 128 R2 (signé) = −128

2.
```
mov   r2, #254
cmp   r2, #0xfe
```

254 → 1111'1110
fe → 1111'1110
0000 0000

1

Z=1 C=1 N=0 V=0 R2(non signé)= ~~0~~ 254 R2(signé) = ~~0~~ −2

3.
```
mov   r2, #-5
subs  r2, #112
```

5: 00000101 → −5: 1111'1011
112 → 0111'0000
1000'1011 → cà2 ⇒ 0111 0101 = 64 32 16 4 1

2

Z= 0 C=1 N=1 V=0 R2(non signé)= 139 R2(signé) = −117

b) Représentez en hexadécimal sur 32 bits (simple précision) la valeur réelle ci-dessous et donnez le développement (pour rappel : exposant est codé sur 8 bits avec un biais de 127)

-130,75 / 64 : $1000\,0010.11 \cdot 2^{-6} \rightarrow 1.0000\,0101\,1 \cdot 2^{1}$

S = 1

E = 127 + 1 = 128 = 1000'0000

M = 1 100'0 0001 0000 0000 0000

2

resultat: 0xC002C000

c) Citez les fanions (flags) utilisés pour tester les conditions des nombres signés et non-signés et indiquez l'équation logique sur les fanions pour les opérations « eq » et « hi ».

non-signés : C et Z ✓

eq → Z=1, C=1 → Z ET C ⇔ Z·C (car on test avec subs et le fanion est init à 1)

hi → Z=0, C=1 → $C \cdot \bar{Z}$ ✓

1/1

signés : C, Z et N ✓

~~eq → Z=1, N=0, C=1 → Z · N̄ · C~~

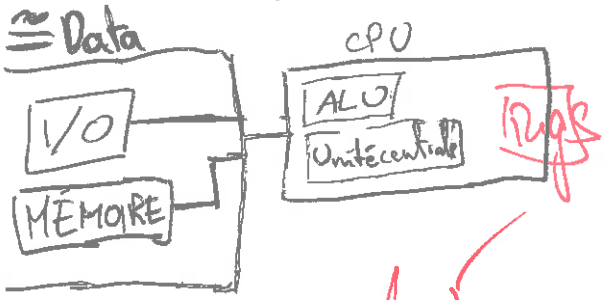
~~hi → N=0, Z=0, C=1 → N̄ · Z̄ · C~~

Systèmes Embarqués 1 : Travail écrit no 1.

9

Problème n° 4 (architecture générale)

a) Décrivez à l'aide d'une figure l'architecture générale des systèmes à microprocesseurs et citez les 3 composantes de l'unité centrale.



1.5

Unité centrale comprend:

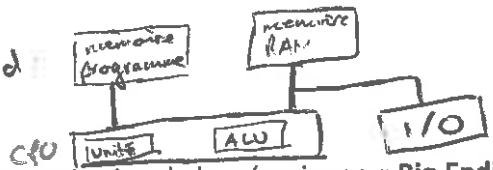
b) Citez les 2 architectures fondamentales des microprocesseurs SISD selon la classification de Flynn et indiquez leur(s) différence(s) principale(s).

Von Neumann :

✓

3

Harvard



✓

c) Pour une organisation de la mémoire en « **Big-Endian** », représentez (en hexadécimal pour les entiers et en caractère ascii pour les strings) dans le tableau ci-dessous les variables suivantes

Adresse :	variable :	taille/type :	valeur :
0xb003210a	text:	.asciz	"some" ✓
0xb0032103	code:	.byte 8	-4_{10} → 0100 → 1011+1 1100 → 0xC
0xb0032108	crc:	.short 16	513_{10} → '0010000'0001
0xb0032104	val:	.long 32	$08723a_{16}$
0xb0032100	parity:	.short 16	2005_8 → 0000 0101 000 000 101 → 0x0405

	7 … 0
0xb0032100	0x04
0xb0032101	0x05 ✓
0xb0032102	
0xb0032103	0x0C
0xb0032104	0x00
0xb0032105	0x08
0xb0032106	0x72
0xb0032107	0x3a
0xb0032108	0x02
0xb0032109	0x01 ✓
0xb003210a	'S'
0xb003210b	'O'
0xb003210c	'M'
0xb003210d	'E'
0xb003210e	'\0' ✓

4,5

Systèmes Embarqués 1 : Travail écrit no 1.

6

Problème n° 5 (architecture interne)

a) Citez ou dessinez les composants principaux de la structure interne des processeurs ARM9

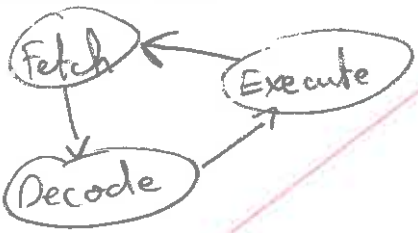
- adress register (et @ incrementer)
- bank register
- ALU (avec MAC et Barrel Shifter)
- IR (Decode Stage et Instruction Decoder
- READ/Write Data register

3

b) Décrivez succintement la fonction/l'usage des différents registres du mode d'opérations « Supervisor »

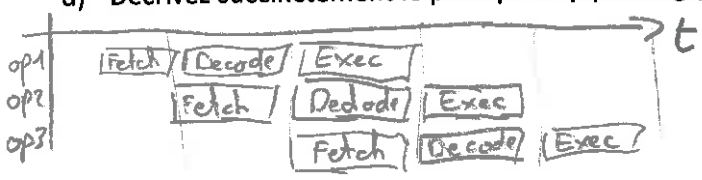
0

c) Décrivez le principe de fonctionnement du processeur RISC i.MX27 (ARM9) le distinguant des µPs CISC



0,5

d) Décrivez succinctement le principe de pipelining (problématique, solution, résultat)



2

Problème : - si op1 a une erreur de décode, doit att pour re-fetch
- le bruit sur le bus