Microprocesseurs 1 & 2: Travail écrit no 1.

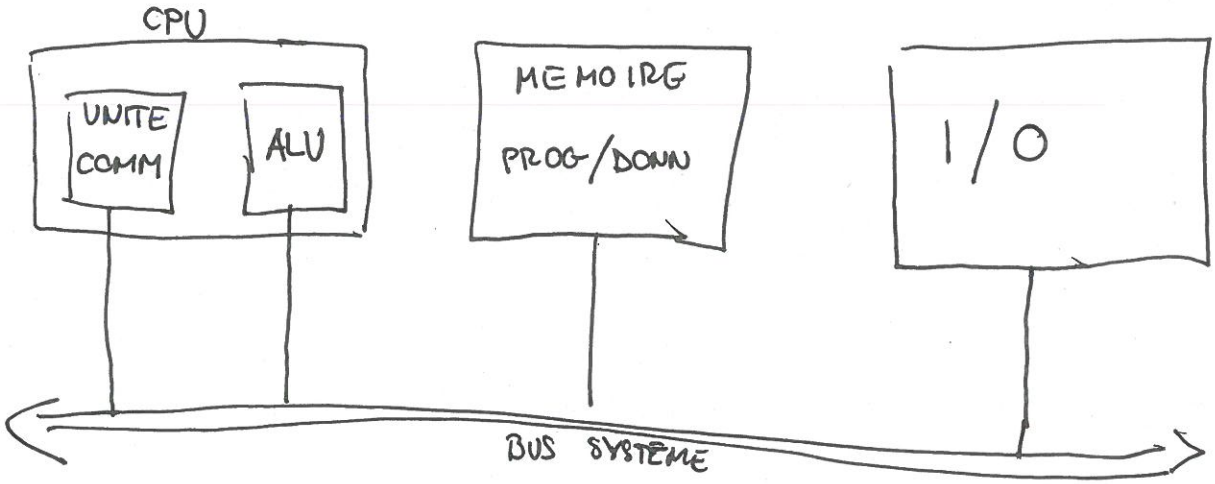
Nom : CASTELLETTI **Prénom :** ALESSIO **Classe : I/2** **Date : 21.11.2011**

8 4.4

Problème n° 1 (architecture générale)

a) Dessinez et décrivez succinctement l'architecture Von Neumann:

4.1



explications.

b) Pour une organisation de la mémoire est en « **little-endian** », représentez (en hexadécimal pour les entiers et en caractère ascii pour les strings) dans le tableau ci-dessous les variables suivantes :

3.1

	Adresse :	variable :	taille/type :	valeur :	
1	0xa0000100	var1	.asciz	"bonjour"	→ ✓
1	0xa000010d	var2	.byte 8	252_{10}	→ 0x FC ✓
0	0xa0000110	var3	.long 32	$67832b_{16}$	→ 0x 67832b ✓
1	0xa0000116	var4	.short 16	406_8	→ 0x 106 ✓
0	0xa000010a	var5	.byte 8	-4_{10}	→ 0x FFFFFFFC

0.11

15 … 8	7 … 0	
0	B	0xa0000100 ← 16 bit
J	N	0xa0000102
U	O	0xa0000104
Ø	R	0xa0000106 ✓
		0xa0000108
0x FFFF	0x FFFC	0xa000010a ←
FC		0xa000010c
		0xa000010e
0x 00 83	0x 67 26	0xa0000110 ←
0x 2b 0	0x 83 67	0xa0000112
		0xa0000114
0x 01	0x 06 ✓	0xa0000116 ←
		0xa0000118

Microprocesseurs 1 & 2: Travail écrit no 1.

10

Problème n° 2 (architecture interne)

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

5

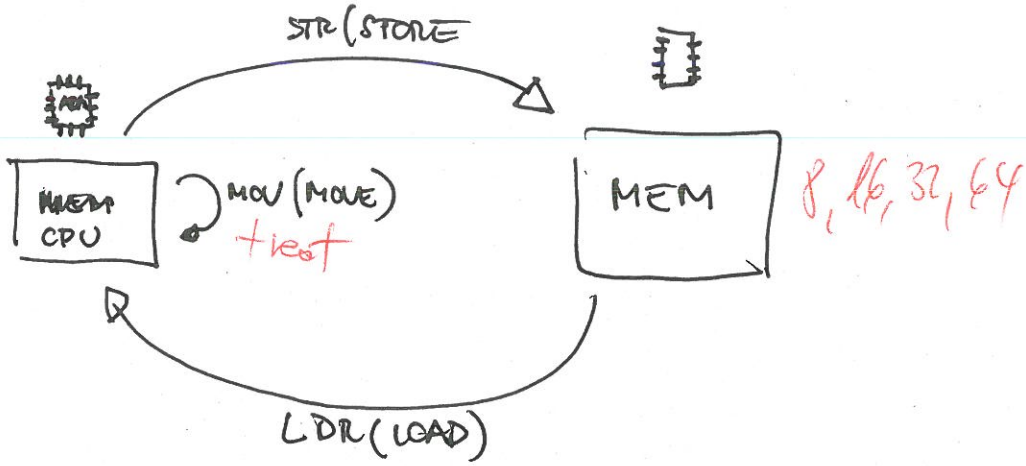
- BARREL / SHIFTER
- MAU
- ALU
- BANK REGISTER
- IR/DECODE
- CONTROL LOGIC

- R/W (REGISTER)
- Adder (REGISTER + INVERTER)

b) Décrivez (avec un graphique ou une figure) le principe de fonctionnement des processeurs ARM:

4,5

Le processeur ARM [illegible] en modalité "load & Store"



On peut dire qu'on traite les données depuis le registre de la mémoire. Pour chaque gestion nécessite d'interagir avec des données dans la mem. on charge une load et on remet une STR (STORE)

Microprocesseurs 1 & 2: Travail écrit no 1.

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 R0 (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   r0, =247
   cmp   r0, #-9
```

2

1111 0111 − 1111 0111 = 0000 0000

Z= 1 C= 1 N=0 V=0 R0(signé)= −9 R0(non signé) = 247

```
2. mov   r0, #139
   adds  r0, #193
```

2

1000 1011 + 1100 0001 = (1) 0100 1100 (76_{10})

Z= 0 C= 1 N=0 V= 1 R0(signé)= 76 R0(non signé) = 76

```
3. mov   r0, #-3
   subs  r0, #125
```

2

1111 1101 − 0111 1101 = 1000 0000

Z= 0 C= 1 N=1 V= 0 R0(signé)= −128 R0(non signé) = 128

b) Représentez en hexadécimal sur 32 bits (simple précision) les valeurs réelles suivantes et donner le développement :

(pour rappel : exposant est codé sur 8 bits avec un biais de 127)

1,5 a) 45,375 : 0x 425A1800 4235'8000

1,1 b) 49 / 2048 ($= 2^{-11}$) : 0x 3CE20000 3CC4'0000

a) 45 → 101101 ✓
,375 → 0011 ? 0,375 = 011
→ 101101,0011 $\cdot 2^0$ = 1,01101 0011 $\cdot 2^5$
E = 127 + 5 = 132 ✓ → 1000 0100 ✓
S = 0 ✓

0 1000 0100 | 0010 1 101 | 1010 | 0001 | 1000 ...
4 2 5 A 1 8

b) 49 → 110 001 ✓
2048 → 2^{11} ✓
E = 127 − 6 = 121 ✓ → 0111 1001
$110001 \cdot 2^{-11} = 1,10001 \cdot 2^{-6}$ ✓
S = 0

0 0111 1001 | 1000 1 | 0010 | 0000 | 0000 ...
3 C E 2 0 0

Microprocesseurs 1 & 2: Travail écrit no 1.

Problème n° 4 (Mode d'adressage)

Pour le code assembleur et la représentation de la mémoire (little-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 (little-endian / 8 bits)

0xa0001000	0x43
0xa0001001	0x83
0xa0001002	0x97
0xa0001003	0x25
0xa0001004	0xd7
0xa0001005	0x25
0xa0001006	0x73
0xa0001007	0xc2
0xa0001008	0xaa
0xa0001009	0x89
0xa000100a	0x00
0xa000100b	0xc0
0xa000100c	0xF6
0xa000100d	0xFF
0xa000100e	
0xa000100f	

Registres (avant)	
R0	0xa000'0100
R1	0x0000'1022
R2	0x0000'0400
R3	0xff00'ff00
R4	0xa000'1000
R5	0x0000'0001
R6	0x0000'0004
R7	0xffff'8ff6
R8	0xa000'1008
R9	0x0000'0100
R10	0x0000'0000
R11	0xa000'0100
R12	0x0000'0010

Registres (après)

0xa000 5022
← 0x0000 0107
← 0x000 0105
0xa000 1005
← 0x000 0106
← 0x000 0107
← 0x000 011C
← 0x0 0e 111
0xa000 1108
← 0x 0000112

0. `0xa000'4400: backup: .long 102,105,106,107,110,111,112`

1. `add r0,r1,r2,lsl #4`

2. `ldrb r3, [r4,r6]`

3. `strh r7, [r4,#12]!`

4. `ldr r1, [r8],r12,lsl #4`

5. `ldr r9, =backup`
   `ldmia r9!, {r2,r5-r7,r10-r12}`

Microprocesseurs 1 & 2: Travail écrit no 1.

Problème n° 5 (Programmation en assembleur)

Coder en langage assembleur ARM l'algorithme suivant :

```
#define MAX 200
char str[] = "un message avec des chiffres 123458";
char msg[MAX]; long digits=0; short len=0;

void main() {
  int i = 0; int j=0;
  do {
    char c = str[i++];
    if ((c >= '0') && (c <= '9')) {
      digits++; c = '*';
    }
    msg[j++] = c;
    len++;
  } while ((c != 0) && (len < MAX));
  msg[MAX-1] = 0;
}
---en assembleur--------
MAX = 200
str:        .asciz  "un message avec des chiffres 123458"
msg:        .space MAX
digits:     .long 0
len:        .short 0

main:
```

ldr r0, str ldr r6 = len; Rodr r3 ≠ 0 // i = 0
ldr r1, msg ldr r2 [r6] ; ldr r4 ≠ 0 // j = 0

loop ldrb r3 [r0, r3]

cmp [r3, #0]
beq end

add r3, #1 ;
cmp r3, #0 ;
bls cont2

cont2 cmp r3, #9 :