

NOLLE

Damien

L3 – Informatique

ADO – Devoir 2 :

<u>Note :</u>	<u>Observation :</u>
<u>/20</u>	

Exercice 1)

Donnez les valeurs affichées par le programme ci-dessous. Pour évaluer votre niveau de compréhension n'utiliser pas de calculatrice ni d'ordinateur (comme ce sera le cas le jour de l'examen). Les valeurs hexadécimales %x sont composées de chiffres et de lettres minuscules (pas de 0x devant ni de 0 à gauche)

```
float foo(float f){
    unsigned n = *(unsigned *) &f;
    unsigned e= ((n >> 23 ) & 0xFF)-127;
    unsigned v = ((1 << e)-1)<<(23-e);
    printf("%x, %d, %x\n", n, e, v);
    n = (n & 0xFF800000) | (n & v);
    return *(float *) &n;
}
```

```
int main(){
    printf("%f\n",foo(-123.25));
    printf("%f\n",foo(5.125));
    return 0;
}
```

Valeur hexadécimale de n pour -220.75 ✓

Valeur décimale de e pour -220.75 ✓

Valeur hexadécimale de n pour -220.75 ✗

Valeur réelle affiché par main pour -220.75 ✓

Valeur hexadécimale de n pour 5.125 ✓

Valeur décimale de e pour 5.125 ✓

Valeur hexadécimale de n pour 5.125 ✗

Valeur réelle affiché par main pour 5.125 ✓

Que fait cette fonction (non demandé ici)

La première instruction permet de récupérer la représentation IEEE 754 (la représentation mémoire) d'un réel simple précision (float).

-220.75

$220 = 1101\ 1100$

$220 - 128 = 92 - 64 = 28 - 16 = 12 - 8 = 4 - 4 = 0$

$0,75 = 11$

$0.5 + 0.25 = 0.75$

$220,75 = 1101\ 1100,11$

Représentation scientifique :

$1,101110011 * 2^7$

Mantisse : 101110011

$7 + 127 = 134$

$134 - 128 = 6 - 4 = 2 - 2 = 0$

1000 0110

Puisque le nombre est négatif, le bit de signe est à 1.

IEEE 754 :

$1100\ 0011\ 0101\ 1100\ 1100\ 0000\ 0000\ 0000 = 0xC35CC000$

$8 + 4 = 12 \rightarrow C$

$1 + 2 = 3$

$4 + 1 = 5$

$8 + 4 = 12 \rightarrow C$

$$8 + 4 = 12 \rightarrow C$$

0

0

0

$$e = ((n \gg 23) \& 0xFF) - 127$$

$$1100\ 0011\ 0101\ 1100\ 1100\ 0000\ 0000\ 0000 \gg 23 = 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1000\ 0110$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1000\ 0110 \& 0xFF$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1000\ 0110$$

$$\& 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 1111$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1000\ 0110$$

$$1000\ 0110 = 128 + 4 + 2 = 134$$

$$134 - 127 = 7$$

$$1100\ 0011\ 0101\ 1100\ 1100\ 0000\ 0000\ 0000 \& 0xFF800000$$

$$1100\ 0011\ 0101\ 1100\ 1100\ 0000\ 0000\ 0000$$

$$\& 1111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000$$

$$1100\ 0011\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000$$

$$v = ((1 \ll e) - 1) \ll 23 - e$$

$$(1 \ll 7) - 1 = 1000\ 0000 - 1 = 1000\ 0000 + (-1)$$

$$\text{NOT}(1) + 1 = \text{NOT}(0000\ 0001) + 1 = 1111\ 1110 + 1 = 1111\ 1111$$

$$1000\ 0000 + 1111\ 1111 = 0111\ 1111$$

$$0111\ 1111 \ll 23 - e = 0111\ 1111 \ll 23 - 7 = 0111\ 1111 \ll 16 = 0111\ 1111\ 0000\ 0000\ 0000\ 0000$$

$$1100\ 0011\ 0101\ 1100\ 1100\ 0000\ 0000\ 0000 \& 0111\ 1111\ 0000\ 0000\ 0000\ 0000$$

1100 0011 0101 1100 1100 0000 0000 0000
& 0000 0000 0111 1111 0000 0000 0000 0000
0000 0000 0101 1100 0000 0000 0000 0000

1100 0011 0000 0000 0000 0000 0000 0000
| 0000 0000 0101 1100 0000 0000 0000 0000
1100 0011 0101 1100 0000 0000 0000 0000

C35C0000

$$1000\ 0110 = 128 + 4 + 2 = 134 - 127 = 7$$

$$1,10111 * 2^7 = 1101\ 1100 = 128 + 64 + 16 + 8 + 4 = 192 + 28 = -220$$

5.125

$$5 = 101$$

$$0,125 = 001$$

101,001

$$1,01001 * 2^2$$

$$2 + 127 = 129 = 1000\ 0001$$

$$0100\ 0000\ 1010\ 0100\ 0000\ 0000\ 0000\ 0000 = 0x40A40000$$

4

0

$$8 + 2 = 10 \rightarrow A$$

4

0

0

0

0

$e = ((0100\ 0000\ 1010\ 0100\ 0000\ 0000\ 0000\ 0000 \gg 23) \& 0xFF) - 127$

0000 0000 0000 0000 0000 0000 1000 0001
& 0000 0000 0000 0000 0000 0000 1111 1111
0000 0000 0000 0000 0000 0000 1000 0001

$129 - 127 = 2$

$v = ((1 \ll 2) - 1) \ll (23 - 2)$

$100 - 1 = 100 + (-1)$

$\text{Not}(1) + 1 = 110 + 1 = 111$

$100 + 111 = 011$

$011 \ll 21 = 0110\ 0000\ 0000\ 0000\ 0000\ 0000$

$n = (0100\ 0000\ 1010\ 0100\ 0000\ 0000\ 0000\ 0000 \& 0xFF800000) \mid (0100\ 0000\ 1010\ 0100\ 0000\ 0000\ 0000\ 0000 \& 0110\ 0000\ 0000\ 0000\ 0000\ 0000)$

0100 0000 1010 0100 0000 0000 0000 0000
& 1111 1111 1000 0000 0000 0000 0000 0000
0100 0000 1000 0000 0000 0000 0000 0000

0100 0000 1010 0100 0000 0000 0000 0000
& 0000 0000 0110 0000 0000 0000 0000 0000
0000 0000 0010 0000 0000 0000 0000 0000

```

0000 0000 0010 0000 0000 0000 0000 0000
| 0100 0000 1000 0000 0000 0000 0000 0000
0100 0000 1010 0000 0000 0000 0000 0000

```

$$129 - 127 = 2$$

$$1,01 * 2^2$$

$$101 = 4 + 1 = 5$$

$$0100\ 0000\ 1010\ 0000\ 0000\ 0000\ 0000\ 0000 = 0x40A0\ 0000$$

Cette fonction permet de calculer la partie entière d'un nombre réel simple précision (float), à partir du nombre au format IEEE 754.

Exercice 2)

Q1 :

1. Codage des instructions Donnez les valeurs hexadécimales 32 bits représentant les instructions 1 à 5 du code ci-dessous:

```

count:
    addu $v0,$0,$0
countBoucle:
    beq $a0,$0,countFin # inst1
    andi $at,$a0,1 #inst2
    add $v0,$v0,$at #int3
    srl $a0,$a0,1 #inst4
    beq $0,$0,countBoucle #inst5
countFin:
    jr $ra

```

#inst1 0x	10800004	#inst2 0x	30810001	#inst3 0x	00411020
#inst4 0x	00042042	#inst5 0x	1000ffff		

beq \$a0, \$0, countFin

C0 = 4 = 0001 00

rs = \$a0 → \$4, 4 = 0 0100 (5bits)

rt = \$0, 0 = 0 0000

countFin : → 4 = 0100

(cp + 4) + (immS16 SLL 2)

4 + 4 = 8

4 SLL 2 = 0001 0000 = 16

16 + 8 = 24

0001 0000 1000 0000 0000 0000 0000 0100

0x10800004

andi \$at, \$a0, 1

C0 = 0xC = 00 1100

rt → \$at → \$1 = 0 0001

rs → \$a0 → \$4, 4 = 0 0100

immNs16 → 1 = 0000 0000 0000 0001

0011 0000 1000 0001 0000 0000 0000 0001

0x30810001

add \$v0, \$v0, \$at

C0 = 00 0000

rd → \$v0 → \$2 = 0 0010

rs → \$v0 → \$2 = 0 0010

rt → \$at → \$1 = 0 0001

shamt = 0 0000

Nf = 10 0000

0000 0000 0100 0001 0001 0000 0010 0000
0x00411020

srl \$a0, \$a0, 1
C0 = 00 0000
rd → \$a0 → \$4, 4 = 0 0100
rt → \$a0 → \$4, 4 = 0 0100
shamt → 0 0001
Nf = 00 0010

0000 0000 0000 0100 0010 0000 0100 0010
0x00042042

beq \$0 \$0, countBoucle
Co = 4 = 0001 00
rs ← 0 0000
rt ← 0 0000
ImmS16 = -5 = 1111 1111 1111 1011

0001 0000 0000 0000 1111 1111 1111 1011
0x1000FFFB
 $8+2+1 = 11 \rightarrow B$

$4/4 - (20 + 4) / 4 = 1 - 6 = -5$

$(Cp + 4) + (ImmS16 \text{ SLL } 2)$

$20 + 4 = 24$

5 = 0000 0000 0000 0101
NOT(0000 0000 0000 0101) + 1

1111 1111 1111 1010 + 1 = 1111 1111 1111 1011

1111 1111 1111 1011 SLL 2 = 1111 1111 1110 1100

24 + -20 = 24 - 20 = 4

Q2:

2. Complétez le code assembleur de la fonction **foo1** du QCM 1

```
unsigned foo1(unsigned f,unsigned m,unsigned w){
    unsigned mb = ~m;
    if (f==1)
        return w|m;
    else
        return w&mb;
}
```

foo1:

#\$a0 -> f, \$a1 -> m, \$a2 -> w

#\$v0 résultat

beq	✓	\$a0	✓	,\$0,foo1Suite
or	✓	\$v0	✓	,\$a1, \$a2
jr	✓	\$ra	✓	

foo1Suite:

nor	✓	\$v0	✓	,\$0, \$a1
and	✓	\$v0	✓	,\$v0, \$a2

idem instruction avant foo1Suite

4 = 0100

0 = 0000

nor 1011

Q3 :

3. Complétez le code assembleur de l'appel fonction foo1 et de l'affichage du résultat en hexadécimal.

```
.data
N1 : .byte -0x22
N2 : .byte 111
res : .byte

.text
# Paramètres foo1(1,N2,N1)
addiu $a0, $0, 1
lbu $a1, 0x2001($0)
lbu $a2, 0x2000($0)
jal foo1

#Afficher le résultat en hexadécimal
addu $a0, $0, $v0
addiu $v0, $0, 34
syscall
# Fin du programme
```

Exercice 3)

4. Exceptions et entrées/sorties. L'instruction break lève une exception ayant une cause valant 9. Complétez le code affichant le message Instruction Break sur la console externe.

```
.kdata
msg : .asciiz "Instruction break"
.text
addiu $a0, $0, 1
break
addi $v0, $0, 10
syscall
.ktext 0x4180
mfc0 $k0, $13
srl $k0, $k0, 2
andi $k0, $k0, 1f
addiu $k1, $0, 9
bne $k0, $k1, pasBreak
addiu $k1, $0, 6000
BreakLoop:
lbu $k0, 0($k1)
beq $k0, $0, pasBreak
sb $k0, 0x7f0c($0)
addiu $k1, $k1, 1
beq $0, $0, BreakLoop
pasBreak:
addiu $v0, $0, 10
syscall
```