TP 2 : Organisation Des Données En Mémoire - Instructions Mémoire

Prénom : _____ **Nom :** _____

Jeux d'instructions - MIPS32

1 Alignement mémoire

Soit la déclaration de variables C suivante :

```
char   X[9] = { 0x10, 0x32, 0x54, 0x76, 0x98, 0xBA, 0xDC, 0xEF, 0x01 };
short  Y[2] = { 0x1234, 0x5678 };
int    Z    = 0xABCDEFAC;
float  A    = 1.5;
double B    = 1.5;
int    C    = 10;
char   D[]  = "hello world";
char   E[]  = "fin de l'exercice";
```

Cette déclaration correspond à la zone `.data` du programme `mips32_align.s` :

```
.data
X : .byte  0x10, 0x32, 0x54, 0x76, 0x98, 0xBA, 0xDC, 0xEF, 0x01
Y : .half 0x1234, 0x5678
Z : .word 0xABCDEFAC
A : .float 1.5
B : .double 1.5
C : .word 10
D : .asciiz "hello world"
E : .asciiz "fin de l'exercice"
```

Avec le simulateur `xspim` ou `QtSpim`, après chargement du programme, observer le contenu mémoire dans la zone `.data` (user data segment) en utilisant l'exécution pas à pas.

1.1 Quelles sont les adresses des différentes variables `X` à `E` ? En déduire les règles d'alignement.

2 Big endian et little endian - Implémentation mémoire

Big endian	Octect 0 MSB	Octect 1	···	Octect N LSB
Little endian	Octect N MSB	Octect N - 1	···	Octect 0 LSB

2.1 En observant l'implémentation mémoire du programme `mips32_align.s`, peut-on en déduire la nature big endian ou little endian pour MIPS32 ?

3 Instructions mémoire

Soit le programme `mips32_reg.s` dont la section `.data` est la suivante :

```
.data
X: .byte 0x10, 0x32, 0x54, 0x76, 0x98, 0xBA, 0xDC, 0xEF, 0x01
Y: .word 0x76543210, 0xEFDCBA98
```

3.1 Quels sont les contenus des registres après l'exécution des instructions suivantes :

```
la $t0, X
lw $t1, 0($t0)
lw $t2, 8($t0)
lb $t3, 5($t0)
lh $t4, 2($t0)
lhu $t5, 6($t0)
lbu $t6, 3($t0)
la $t0, Y
lw $t7, 0($t0)
lw $s0, 4($t0)
jr $ra
```

4 Suite de Fibonnaci

Le programme `mips32_fibonnaci.s` range dans le tableau d'entiers 32 bits `X` les deux premières valeurs de la suite de FIBONNACI.

4.1 Sans utiliser de boucle, compléter le programme pour écrire les 4 valeurs suivantes. Écrire une variante qui rangera les 6 valeurs dans un tableau d'octects.

Jeux d'instructions - ARM

5 Implémentation mémoire

Le programme `arm_align.s` a la zone donnée suivante :

```
.data
A: .byte 0x39, 0x32, 0x24, 0xAB, 0xDA
B: .word 0x98765432, 0xFA00, 0xFEDCBA98
C: .asciz "Hello world"
D: .word 345
```

5.1 Exécuter ce programme pour observer les règles d'alignement mémoire. ARM utilise-t-il big ou little endian ?

6 Instructions mémoire

Exécuter le programme `arm_reg.s` dont la zone donnée est la suivante :

```
.data
A: .byte 0x10, 0X32, 0x54, 0x76, 0x98, 0xBA, 0xDC, 0xEF, 0x01
```

6.1 Quels sont les contenus des registres après exécution du programme suivant :

```
LDR r0,=A @ Chargement adresse de A
MOV r1,#4
LDR r2,[r0,#4]
LDR r3,[r0,#4]!
LDR r4,[r0],#8
LDR r5,[r0,-r1,LSL#1]
SWI 0x11 @ Stop program execution
```

7 Suite de Fibonnaci

Le programme `arm_fibonnaci.s` range dans le tableau d'entiers 32 bits `X` les deux premières valeurs de la suite de Fibonnaci.

7.1 Sans utiliser de boucle, compléter le programme pour écrire les 4 valeurs suivantes. Écrire une variante qui rangera les 6 valeurs dans un tableau d'octects.