2018-2019

TP 2 - Mémoire et boucles

On utilise le simulateur QtSpim. Les programmes sont disponibles sur le site du cours.

1. Endianité

Charger le programme tp2_code1.s. Afficher l'onglet Data. La colonne la plus à gauche affiche la première de 16 adresses. La zone la plus à droite affiche l'interprétation ASCII des 16 octets correspondants.

a. Exécuter le programme. Donner le contenu en hexadécimal des registres $t0 et $t1 après l'exécution.

$t0	
$t1	

b. Décoder l'instruction correspondant à la pseudo-instruction la $t0, X et expliquer son action.

La $t0, X	

Expliquer son action.

Décommenter la ligne lw $t1, 1($t0) et réexécuter le programme. Que constate-t-on ?

Explication.

c. Charger le code tp2_code2.s. et l'exécuter. Ecrire ci-dessous le contenu de la mémoire à partir du premier octet de la variable X.

Adresse	**Contenu**
0x10010000	**…**

d. QtSpim utilise l'endianité de la machine sur laquelle il s'exécute. D'après les effets des instructions précédentes, quelle est l'endianité de votre machine (justifier) ?

e. **Retour au premier code :** à quelle adresse se trouve le caractère 'Q' dans la zone *User data segment*?

x

f. Compléter le programme en utilisant **une** instruction de rangement pour que la zone la plus à droite affiche «ArchitectureL2S3 ».

Instruction utilisée et valeurs des registres concernés :

2. Boucle for :

Coder les instructions C suivantes (en remplissant le fichier tp2_code3.s) :

```
int X[10] ;
int Y[10] ;
int Z[10] ;
for(int i=1 ; i<10 ; i++)
    Z[i] = X[i-1] + y[i] ;
```

3. Boucle

Un pseudo-code non optimisé du tri bulle est donné en figure 1.

```
procedure bubbleSort( A : list of sortable items )
  n = length(A)
  repeat
    swapped = false
    for i = 1 to  n-1 inclusive do
      if A[i-1] > A[i] then
```

```
      swap( A[i-1], A[i] )
      swapped = true
    end if
  end for
 until not swapped
end procedure
```

Figure 1: tri bulle

Le programme tp2_code4.s contient un squelette de programme MIPS32 implémentant le corps de cette procédure.

a. Charger un programme ne contenant que les trois premières lignes (les lignes qui chargent les données donc). Expliquer le contenu de la zone *User data segment*.

b. Quel doit être le contenu de cette zone après le tri ?

c. Compléter le programme et vérifier qu'il donne un résultat correct.

Etiquettes	**Instructions**	**Commentaires**
main :		
deb :	xor $t4,$t4,$t4	#swapped = false
	la $t3, X	#initialisation boucle
	addi $t5, $t3, 4	
blc :	REMPLIR	#t1 = A[i-1]
	REMPLIR	#t2 = A[i]
	slt $t6, $t2, $t1	
	REMPLIR	#pas d'échange à faire
	REMPLIR	#A[i-1] <-> A[i]

	REMPLIR	#A[i-1] <-> A[i]
	move $t4, $t6	#swapped=true
suite:	REMPLIR	#i++
	bne $t3, $t5, bcl	#retour début boucle for
	REMPLIR	#retour début boucle do-while
	jr $ra	

Essais effectués :
