centre de télé-enseignement universitaire
formations à distance à votre rythme par e-learning

ctu UNIVERSITÉ DE FRANCHE-COMTÉ UFC

Licence d'informatique - Année 2017/2018

Architecture des Ordinateurs

Examen de Mai 2018 - Durée 3 heures

Documents autorisés : aucun - pas de calculatrice

- Si vous détectez une erreur de sujet, merci de l'indiquer dans votre copie.
- Pour être valables, vos réponses doivent être justifiées sur votre copie.
- Complétez et joignez la feuille de réponse à votre copie
- Utilisez les noms des variables et des fonctions proposées dans l'énoncé.
- Les questions sont indépendantes.

Exercice 1 : Représentation des nombres

```
1  void main(){
2      int N1= 0x23456789,N2=0x98765432;
3      int N3 = Float.floatToIntBits(-8.125f);
4      //en C  x=-8.125f ; unsigned N3= *(unsigned*) & x;
5      printTabChar(toHexString(N3));
6      printTabChar(toHexString(N1&N2));
7      printTabChar(toHexString(N1|N2));
8      printTabChar(toHexString(N1^N2));
9      printTabChar(toHexString(N2>>1));
10     printTabChar(toHexString(N2>>>1)); // EN C (unsigned)N2 >>1
11     printTabChar(toHexString(foo(N1,N2)));}
12 static int foo(int x,int y){
13         int res=0;
14         int i;
15         for (i=0;i<4;i++){
16              res = (res >> 8 )  & 0x00FFFFFF; // Ou (res >>> 8) en Java
17             res = res | (x << 28) | ((y & 0xF )<<24);
18             x = x >> 4;
19             y = y >> 4
20         }
21         return res;
22     }
```

Question 1 : Expliquez et donnez, sur la feuille de réponse, les valeurs hexadécimales affichées par les lignes 5 à 10 du programme ci-dessus

Question 2 : Expliquez et donnez, sur la feuille de réponse, la valeur hexadécimale affichée par la ligne 11 du programme ci-dessus. Que fait la fonction *foo* ?

Exercice 2 : Assembleur Mips

Complétez, sur la feuille de réponse, le code de la fonction *foo* de l'exercice précédent et de son appel avec les valeurs du programme précédent. Le code placera le résultat de la fonction dans la variable *res* et l'affichera en hexadécimal.

```
#Le programme assembleur sera utilisé pour
    les exercices suivants
# initialement $a0 = 0, $a1=32
# $v0=0, cp = 0x100
testCache:
  lw  $t0,0x2000($a0) # instruction 1
  add  $v0, $v0, $t0 # instruction 2
  lw  $t1,0x2100($a0)
  add  $v0, $v0, $t1
  lw  $t2,0x2200($a0)
  add  $v0, $v0, $t2
  addi  $a0, $a0,4
  addi  $a1, $a1,-1
  bne  $a1, $zero,testCache # instruction 3
  jr  $ra
```

Exercice 3 : Hiérarchie mémoire

Soit un cache pédagogique de 256 octets, organisé par blocs de 32 octets. Nous souhaitons évaluer les performances du programme ci-dessus. Nous évaluerons les fonctions de correspondance directe puis associative par ensemble de 4 blocs.

Question 1 : Que fait le programme ci-dessus ? Combien d'itérations sont effectuées ?

Question 2 : Expliquez et donnez, sur la feuille de réponse, la taille en bit de l'étiquette, de l'index et du déplacement dans les deux cas.

Question 3 : Expliquez et donnez, sur la feuille de réponse, les taux de succès lors de l'accès au cache. Pour cela, complétez les valeurs hexadécimales des trois champs et du succès d'accès au cache (S ou D) pour les deux premières itérations.

Question 4 : Quel serait ce taux pour une fonction associative par ensemble de 2 ?

Exercice 4 : Pipeline

Question 1 : Indiquez les dépendances lecture après écriture du programme ci-dessus.

Question 2 : Expliquez ce qu'est le branchement retardé d'un cycle.

Question 3 : Complétez, sur la feuille de réponse, le diagramme temporel.

Question 4 : Réorganisez, sur la feuille de, le programme pour ne pas avoir de cycle d'attente avec l'envoi et le branchement retardé d'un cycle.

Exercice 5 : Architecture

Question 1 : Expliquez et donnez, sur la feuille de réponse, les valeurs hexadécimales représentant les instructions 1 à 3 du programme ci-dessus.

Question 2 : Expliquez et donnez, sur la feuille de réponse, les valeurs numériques des signaux lors du traitement des trois instructions précédentes à la première itération du programme. La valeur mémoire en **0x**2000 contient la valeur **0x**10.

Question 3 : Nous souhaitons ajouter l'instruction ***slt rd,rs,rt*** qui effectue l'opération rd =1 si $rs < rt$, rd =0 sinon. Le numéro de fonction de cette instruction est **0x**2A.

1. Quelle opération bit à bit doit être effectuée pour calculer la valeur de rd ? Proposez une modification de l'UAL pour effectuer cette opération.
2. Que devons nous modifier dans le circuit de commande ?
3. D'autres changements sont-ils nécessaires pour implémenter cette instruction ?

Exercice 6 : Exceptions et Entrées/sorties

```
  .data
  x :  .word 0x7FFFFFFE # en 0x2000
.text
  addi  $t0, $zero,0x2000
  lw  $a0,3($t0)
  addi  $v0, $zero,34
  syscall
  addi  $v0, $zero,10
  syscall
```

Question 1 : Quelle valeur devrait être affichée par le programme ci-dessus ?

Question 2 : Expliquez ce qui pose problème lors de l'exécution de ce programme et corrigez le pour afficher la valeur présente dans la mémoire **0x**2003 ?

Feuille de réponse à joindre à votre copie sans nom ni numéro d'étudiant

toHexString(N3)	
toHexString(N1& N2)	
toHexString(N1 | N2)	
toHexString(N1 ^ N2)	
toHexString(N2 >> 1)	
toHexString(N2>>> 1)	
toHexString(foo(N1,N2))	

```
.data
  N1 :  .word  0x23456789
  N2:  .word 0x98765432
  res :  .word 0
.text
   lw  $a0,_______(_______)
   lw _______,_______(_______)
   jal  foo
   sw _______,_______(_______)
   add _______, $zero , $v0
   addi  $v0, $zero ,34
   syscall
   addi  $v0, $zero ,_______
   syscall
foo:
# $a0  et  $a1   param  $v0
   res
   sw  $t0 ,0( $sp)
   addi  $sp, $sp ,_______
   sw  $t1 ,0( $sp)
   addi  $sp, $sp ,_______
   addi  $t0, $zero ,_______
   addi  $v0, $zero ,_______
fooBoucle:
   beq  $t0, $zero ,fooFin
   srl  $v0,_______,_______
   sll  $t1, $a0,_______
   or  $v0, $v0,_______
   andi  $t1, $a1,_______
   sll  $t1, $t1,_______
   or  $v0, $v0,_______
   sra  $a0, $a0,_______
   sra  $a1, $a1,_______
   addi  $t0, $t0,_______
   beq  $zero ,
     $zero ,fooBoucle
fooFin:
   addi  $sp, $sp,_______
   lw _______,0( $sp)
   addi  $sp, $sp,_______
   lw _______,0( $sp)
   jr _______
```

Fonction	Taille Étiquette	Taille Index	Taille Déplacement
Directe			
Associative ensemble 4			

Directe	Étiq	Index	Dép	Succès
0x2000				
0x2100				
0x2200				
0x2004				
0x2104				
0x2204				

Taux de succès : ——

Assoc ens	Étiq	Index	Dép	Succès
0x2000				
0x2100				
0x2200				
0x2004				
0x2104				
0x2204				

Taux de succès : ——

Inst / Cycle	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
lw *$t0*,0(*$a0*)	LI	DI	EX	M	ER										
add *$v0*,*$v0*,*$t0*	X														
lw *$t1*,0(*$a0*)	X	X													
add *$v0*,*$v0*,*$t1*	X	X	X												
lw *$t2*,0(*$a0*)	X	X	X	X	X										
add *$v0*,*$v0*,*$t2*	X	X	X	X	X	X									
addi *$a0*,*$a0*,4	X	X	X	X	X	X	X	X							
addi *$a1*,*$a1*,-1	X	X	X	X	X	X	X	X	X						
bne *$a1*,*$zero*,tes	X	X	X	X	X	X	X	X	X	X	X				
lw *$t0*,0(*$a0*)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	LI

Diagramme avec envoi et branchement au plus tôt

```
# Programme réorganisé
 lw   $t0 ,0x2000( $a0)
_____________________
_____________________
_____________________
 add   $v0 , $v0 , t1
_____________________
 add   $v0 , $v0 , t2
 bne   $a1 , $zero , testCache
_____________________
```

Instruction	Codage instruction	num rs	num rt	num ec	ec	res	cp	cp suivant
lw *$t0*,0x2000(*$a0*)								
add *$v0*,*$v0*,*$t0*								
bne *$a1*,*$zero*,testCache								

Architecture

Éléments de correction

toHexString(N3)	**0x**C102 0000
toHexString(N1& N2)	**0x**0044 4400
toHexString(N1 | N2)	**0x**BB77 77BB
toHexString(N1 ^ N2)	**0x**BB33 33BB
toHexString(N2 >> 1)	**0x**CC3B 2A19
toHexString(N2>>> 1)	**0x**4C3B 2A19
toHexString(foo(N1,N2))	**0x**6574 8392 - Entrelacer les 4 digits de poids faibles des deux paramètres

```
.data
  N1 :  .word  0x23456789
  N2:  .word 0x98765432
  res :  .word 0
.text
   lw  $a0,0x2000($zero)
   lw  $a1,0x2004($zero)
   jal foo
   sw  $v0,0x2008($zero)
   add  $a0, $zero, $v0
   addi  $v0, $zero,34
   syscall
   addi  $v0, $zero,10
   syscall
foo:
# $a0 et $a1  param $v0 res
   sw  $t0,0($sp)
   addi  $sp, $sp,-4
   sw  $t1,0($sp)
   addi  $sp, $sp,-4
   addi  $t0, $zero,4
   addi  $v0, $zero,0
fooBoucle:
   beq  $t0, $zero,fooFin
   srl  $v0, $v0,8
   sll  $t1, $a0,28
   or  $v0, $v0, $t1
   andi  $t1, $a1,0xF
   sll  $t1, $t1,24
   or  $v0, $v0, $t1
   sra  $a0, $a0,4
   sra  $a1, $a1,4
   addi  $t0, $t0,-1
   beq  $zero, $zero,fooBoucle
fooFin:
   addi  $sp, $sp,4
   lw  $t1,0($sp)
   addi  $sp, $sp,4
   lw  $t1,0($sp)
   jr  $ra
```

Calcule de la somme des valeurs contenues dans 3 tableaux de 32 entiers situés en **0x**2000, **0x**2100 et **0x**2200.

Fonction	Taille Étiquette	Taille Index	Taille Déplacement
Directe	7	3	5
Associative ensemble	9	1	5

Directe	Étiq	Index	Dép	Succès
0x2000	0x20	0	0	D
0x2100	0x21	0	0	D
0x2200	0x22	0	0	D
0x2004	0x20	0	4	D
0x2104	0x21	0	4	D
0x2204	0x22	0	4	D

Taux de succès : $\frac{0}{96}$

Assoc ens	Étiq	Index	Dép	Succès
0x2000	0x80	0	0	D
0x2100	0x84	0	0	D
0x2200	0x88	0	0	D
0x2004	0x80	0	4	S
0x2104	0x84	0	4	S
0x2204	0x88	0	4	S

Taux de succès : $\frac{84}{96} = \frac{7}{8}$

Inst / Cycle	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
lw $t0,0($a0)	LI	DI	EX	M	ER $t0										
add $v0,$v0,$t0		LI	DI	EX	EX $t0	M	ER $v0								
lw $t1,0($a0)			LI	DI	DI	EX	M	ER $t1							
add $v0,$v0,$t1				LI	LI	DI	EX $v0	EX $t1	M	ER $v0					
lw $t2,0($a0)						LI	DI	DI	EX	M	ER $t2				
add $v0,$v0,$t2							LI	LI	DI	EX $v0	EX $t2	M	ER		
addi $a0,$a0,4									LI	DI	DI	EX	M	ER	
addi $a1,$a1,-1										LI	LI	DI	EX	M $a1	ER
bne $a1,$zero,tes												LI	DI	DI $a1	EX
lw $t0,0($a0)															LI

Diagramme avec envoi et branchement au plus tôt

```
lw   $t0 ,0x2000( $a0)
lw   $t1 ,0x2100( $a0)
lw   $t2 ,0x2200( $a0)
add   $v0 , $v0 ,t0
add   $v0 , $v0 ,t1
addi   $a1 , $a1 ,-1
add   $v0 , $v0 ,t2
bne   $a1 , $zero ,testCache
addi   $a0 , $a0 ,4
```

code optimisé avec envoi et branchement retardé d'un cycle et réorganisation du code

Instruction	Codage instruction	num rs	num rt	num ec	ec	res ual	cp	cp s
lw $t0,0x2000($a0)	**0x**8c88 2000	4	8	8	0x10	0x2000	0x100	0x104
add $v0,$v0,$t0	**0x**0048 1020	2	8	2	0x10	0x10	0x104	0x108
bne $a1,$zero,test	**0x**14A0 FFF7	5	0	?	?	?	0x120	0x100

Architecture

instruction ***slt***

— L'ual doit calculer *rs*-rt >>> 31 qui vaut 1 si $rs < rt$ 0 sinon

— Le circuit de décodage doit décoder la valeur de de Nf =**0x**2A si Co =0

— Rien d'autre a ajouter dans le chemin de données.

Entrées/sorties

Erreur lors de l'accès mémoire **0x**2003, non divisible par 4. Pour résoudre le problème lw $a0,0x2000($zero) puis srl $a0,$a0,24 pour récupérer la valeur **0x**7F présente dans la mémoire octet d'adresse **0x**2003.