

## Examen L2-ADO

09-Janvier-2023

Durée: 90 minutes

Nom:	
Prénom :	/ 20
Groupe:	/ 20
Matricule:	-

## Exercice 1 : (7 points) (0 point pour chaque réponse fausse ou non justifiée)

A/ Remplissez les blancs ci-dessous afin que la fonction **mod16** renvoie le reste de la division de **y** par 16. Le premier blanc doit être un <u>opérateur binaire</u> et le second blanc doit être un nombre décimal.

B/ Donnez le code machine, en hexadécimal, d'une instruction MIPS qui provoquera une boucle infinie si jamais elle est exécutée, n'importe où, dans n'importe quel programme MIPS.

```
Li: beq \$0, \$0, Li # code machine \rightarrow 0x1000ffff (1 pt)
```

C/ En utilisant le moins de lignes vides possible, complétez la version en assembleur MIPS du code C ci-dessous. Vous n'êtes autorisé à utiliser que les registres déjà indiqués dans le texte et les registres \$t0-3 et \$s0-s2 (vous n'aurez pas besoin de les utiliser tous). (5 pts)

```
ISCOMMITOK:
         _<mark>lb</mark>_ $t0 __<mark>0</mark>_($a0)
         _<mark>lb</mark>_ $t1 __<mark>0</mark>_($a1)
                                       (0,5 pt)
  COND: and $t2 $t0 $t1______ (0,5 pt)
         _beq $t2 $0 EXIT______(0,5 pt)
         _bne $t0 $t1 FAILED_____ (0,5 pt)
         addiu $a0 $a0 1
         addiu $a1 $a1 1
         _j ISCOMMITOK_____(0,5 pt)
  EXIT: _or_ $t2 $t0 $t1
         _bne $t2 $0 FAILED______ (0,5 pt)
         li $v0 1
         _j END___(ou jr $ra)____(0,5 pt)
FAILED: li $v0 0
         _jr $ra_____(0,5 pt)
  END:
                           4 lignes vides (0,25 pt)
```

## Exercice 2 : (7 points) (0 point pour chaque réponse fausse ou non justifiée)

Un concepteur de micro-processeur aimerait réaliser une version modifiée de processeur MIPS ayant 16 registres 64-bits au lieu de 32 registres 32-bits. De ce fait, il adapte le jeu d'instructions MIPS pour cette « nouvelle » architecture.

1/ Dans les blancs ci-dessous, spécifiez les tailles des champs pour utiliser au mieux les instructions 32-bits sur cette nouvelle architecture. La taille du champ **opcode** est inchangée. **(2 pts)** 

Type-R:	opcode	rs	rt	rd	shamt	func	
	6	4	4	4	6	8	

2/ Combien d'instructions différentes de type R pouvons-nous avoir dans cette architecture ?

$$2^{\text{taille(func)}} = 2^8$$
 (1 pt)

3/ Dans cette nouvelle architecture, si PC = 0x0AD0, quelle est la plus grande adresse vers laquelle nous pouvons effectuer un saut relatif?  $0x0AD0 + 4 + (2^17-1)\times 4 = 0x0AD0 + 2^19 = 0xAD0 + 0x80000 = 0x80AD0$  (2 pts)

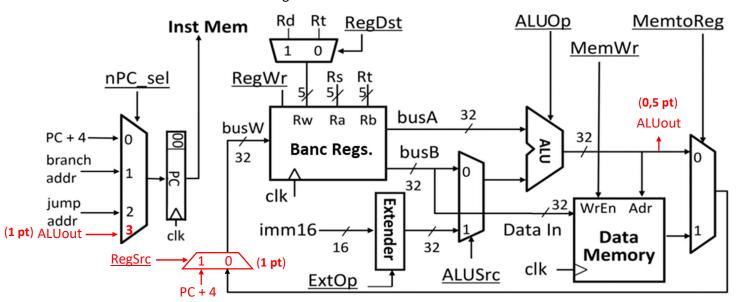
4/ Traduisez le code machine suivant en instruction MIPS dans cette nouvelle architecture (Indiquez des numéros de registre au lieu de noms).

```
0xAE9FFFF8 → 101011 | 1010 | 0111 | 111...1000 # sw $7, -8($10) (2 pts)
```

## Exercice 3: (6 points) (0 point pour chaque réponse fausse ou non justifiée)

Considérez l'instruction suivante : **jals \$rt \$rs imm**. L'instruction stocke PC + 4 dans le registre **\$rt**. En même temps, elle affecte au PC le contenu du registre **\$rs** additionné à une constante signée **imm**.

A/ Modifiez **le moins possible** le chemin de données ci-dessous pour prendre en charge l'instruction **jals**. Dessinez vos modifications directement dans le diagramme et décrivez vos modifications dans le cadre ci-dessous.



Ajouter un mux à busW et ajouter un nouveau signal de commande « RegSrc » pour contrôler ce multiplexeur. Connecter l'entrée busW existante et PC + 4 au nouveau mux. Connecter la sortie ALU au mux contrôlé par nPC\_sel sous le port 3 (nPC\_sel lui-même n'a pas besoin d'être changé car il s'agit déjà d'un signal 2-bits). (1pt)

(3,5 pts)

B/ Définissez les lignes de contrôle pour l'instruction **jals**. Énumérez ce que chaque signal devrait être, soit par un nom intuitif, soit par {0, 1, X, etc.}. Inclure tous les nouveaux signaux de contrôle que vous avez ajoutés. (2,5 pts)

RegDst	RegWr	nPC_sel	ExtOp	ALUSrc	ALUOp	MemWr	MemtoReg	RegSrc	
0	1	3	sign	1	add	0	X	PC + 4	

(0,25 pt) pour chaque réponse correcte.