Département informatique / 2ème Année Licence **NOM** :…… …… ……
Module : Architecture des ordinateurs **PRENOM** :……
Durée : 2H00 / Date : 22 Janvier 2025 **GROUPE** : ……

Examen Final (Répondez sur la feuille)

Questions de compréhension (**5 pts**) **:** Répondre par vrai **(V)** ou faux **(F)** à chacun des énoncés suivants. Une bonne réponse vaut **(+ 0,5 pt)**, une mauvaise réponse vaut (**- 0,5 pt**).

1. Une architecture à accumulation utilise plusieurs registres pour stocker les opérandes.	
2. Les aléas de contrôle sont causés par des instructions dépendantes qui utilisent le même registre.	
3. Le forwarding est une solution aux aléas structurels dans un pipeline.	
4. Une architecture Von Neumann sépare les bus pour les instructions et les données.	
5. Le pipeline permet d'exécuter plusieurs instructions dans le même cycle d'horloge.	
6. Les DRAM sont des mémoires utilisées dans la réalisation des mémoires caches.	
7. Les registres de l'architecture MIPS sont organisés en mémoire principale.	
8. L'utilisation d'un prédicteur de branchement peut réduire les aléas de données dans un pipeline	
9. La largeur du bus d'adresse n'a aucun impact sur la capacité totale d'une mémoire.	
10. Un processeur multi-cycle nécessite plus de matériel qu'un processeur pipeline.	

Exercice n°1 (5 pts) :

Soit l'expression **Z= (A+B/C)*(D+E)**	
Ecrire un programme qui calcule l'expression Z dans une machine possédant **un opérande.** **(machine à une (1) adresses)**	Ecrire un programme qui calcule l'expression Z dans une machine ne possédant pas d'opérande **(machine à zéro (0) adresses)**
Architecture : ……	**Architecture** : ……
……	……

Exercice n°2 (5 pts) :

1. Ecrire le code assembleur MIPS pour le code C ci-dessous? (3 pts)

```
While (X!= Y)
  {
    İf (X < Y)
       { Y = Y – X; }
    Else
       { X = X – Y; }
  }
```

……

Remarques :
- déclarer et initier X et Y en mémoire
- Utiliser les registres **$t1** et **$t2** pour récupérer les valeurs des variables **X** et **Y** à partir de la mémoire.
- Chaque erreur de syntaxe (- 0,25 pt)

2. Que fait ce code ? (2 pts) ……………………………………………………………………………

Exercice n°3 (5 pts) : Programmation MIPS

Soit un tableau T de 10 entiers, avec T(i) ∈ {0, 1}. Ecrire un Programme MIPS qui **retourne la position i** dans le tableau telle que T[i] est le début de la plus longue suite consécutive de zéros.

Exemple : T : 1 0 0 1 1 0 0 0 1 0 > > > > **La position i = 5**

Corrigé Examen Final

Questions de compréhension : Répondre par vrai **(V)** ou faux **(F)** à chacun des énoncés suivants :

1. Une architecture à accumulateur utilise plusieurs registres pour stocker les opérandes.	F
Une architecture à accumulation utilise un seul registre, appelé accumulateur, pour les calculs.	
2. Les aléas de contrôle sont causés par des instructions dépendantes qui utilisent le même registre.	F
Les aléas de contrôle sont causés par des branchements ou des instructions de saut qui modifient le flux d'exécution	
3. Le forwarding est une solution aux aléas structurels dans un pipeline.	F
Le forwarding est une solution aux aléas de données, permettant de transmettre directement un résultat calculé à une instruction en attente sans passer par les registres	
4. Une architecture Von Neumann sépare les bus pour les instructions et les données.	F
Dans une architecture Von Neumann, les instructions et les données partagent le même bus.	
5. Le pipeline permet d'exécuter plusieurs instructions dans le même cycle d'horloge.	F
Le pipeline permet de découper l'exécution d'instructions en étapes qui se chevauchent, mais une seule instruction complète son exécution par cycle	
6. Les DRAM sont des mémoires utilisées dans la réalisation des mémoires caches.	F
Les SRAM sont des mémoires utilisées dans la réalisation des mémoires caches.	
7. Les registres de l'architecture MIPS sont organisés en mémoire principale.	F
Les registres de l'architecture MIPS sont situés dans le processeur, séparés de la mémoire principale.	
8. L'utilisation d'un prédicteur de branchement peut réduire les aléas de données dans un pipeline	F
Il réduit les aléas de contrôle, pas de données	
9. La largeur du bus d'adresse n'a aucun impact sur la capacité totale d'une mémoire.	F
La largeur du bus d'adresse détermine directement le nombre maximum de cases mémoire adressables.	
10. Un processeur multi-cycle nécessite plus de matériel qu'un processeur pipeline.	F
Un processeur pipeline nécessite généralement plus de matériel, car chaque étape du pipeline doit être implémentée avec des registres intermédiaires et des unités fonctionnelles capables de travailler en parallèle.	

Exercice n°1 (5 pts)

Soit l'expression Z= **(A+B/C)*(D+E)**

Ecrire un programme qui calcule l'expression Z dans une machine possédant **un opérande.** **(machine à une (1) adresses)**		Ecrire un programme qui calcule l'expression Z dans une machine ne possédant pas d'opérande **(machine à zéro (0) adresses)**	
Architecture : accumulateur		Architecture : Pile	
LOAD @B; DIV @C; ADD @A; STORE @Z; LOAD @D; ADD @E; MUL @Z; STORE @Z;	[Acc] ←B [Acc] ← B/C [Acc] ←B/C+A [Z] ← A+B/C [Acc] ←D [Acc] ← D+E [Acc] ← (D+E)*(A+B/C) [Z] ← (D+E)*(A+B/C)	PUSH A; PUSH B; PUSH C; DIV; ADD; PUSH D; PUSH E; ADD; MUL; POP Z;	Pile ={A} Pile ={A; B} Pile ={A; B; C} Pile ={A; B/C} Pile ={A+(B/C)} Pile ={A+(B/C); D} Pile ={A+(B/C); D; E} Pile ={A+(B/C); D+E} Pile ={(A+(B/C))*(D+E)} Pile ={ } et [Z] ← (A+B/C)*(D+E)

Exercice n°2 (5 pts)

1. Quel est le code assembleur MIPS pour le code C ci-dessous? (3 pts)

```
While (X!= Y)
  {
    if (X < Y)
       { Y = Y – X; }
    Else
       { X = X – Y; }
  }
```

Remarques :
- déclarer et initier X et Y en mémoire
- Utiliser les registres **$t1** et **$t2** pour récupérer les valeurs des variables **X** et **Y** à partir de la mémoire.
- Chaque erreur de syntaxe (- 0,25 pt)

```
Lw $t1, X
Lw $t2, Y ..... 0,5

While : beq $t1, $t2, Suite_2 ..... 0,5
        bge $t1, $t2, Else ..... 0,5
        sub $t2, $t2, $t1 ..... 0,25
        j Suite_1 ..... 0,25

Else:   sub $t1, $t1, $t2 ..... 0,25

Suite_1 : j while ..... 0,25

Suite_2:
.....
Sw $t1, X
Sw $t2, Y ..... 0,5
```

2. Que fait ce code ? (2 pts)

Le code calcul le PGCD de deux nombres .

Exercice n°3 (5 pts) : Programmation MIPS

Soit un tableau T de 10 entiers, avec T(i) ∈ {0, 1}. Ecrire un Programme MIPS qui retourne la position i dans le tableau telle que T[i] est le début de la plus longue suite consécutive de zéros.

Exemple : T : 1 0 0 1 1 0 0 0 1 0 > > > > La position i = 5

```
# Programme pour trouver la position de début de la plus longue suite consécutive de zéros
# T(i) ∈ {0, 1} et T est un tableau de 10 éléments.

.data
T:  .word 0, 0, 0, 0, 1, 0, 0, 0, 1, 0  # Tableau d'entrée
n:  .word 10                    # Taille du tableau
result: .word -1                # Résultat : position de début de la plus longue suite de 0

.text
.globl main

main:
   # Initialisation des registres
   la $t0, T          # $t0 pointe vers le tableau T
   lw $t1, n          # $t1 contient la taille du tableau (n = 10)
   li $t2, 0          # $t2 : compteur pour parcourir le tableau (index i)
   li $t3, 0          # $t3 : compteur de la longueur courante de zéros
   li $t4, 0          # $t4 : longueur maximale trouvée
   li $t5, -1         # $t5 : position du début de la plus longue suite de zéros courante
zero_case:
   # T[i] == 0
   addi $t3, $t3, 1     # Incrémenter la longueur courante
   beq $t6, -1, set_start
   j check_max

set_start:
   # Définir la position de début de la suite actuelle
   move $t6, $t2

check_max:
   # Si la longueur courante > longueur maximale, mettre à jour la longueur max
   bgt $t3, $t4, update_max
   j next_element

update_max:
   move $t4, $t3       # Mettre à jour la longueur maximale
   move $t5, $t6       # Mettre à jour la position de début

next_element:
   # Passer à l'élément suivant
```

```
    li $t6, -1          # $t6 : position temporaire pour marquer le début de la suite actuelle

loop:
    # Vérifier si on a parcouru tout le tableau
    beq $t2, $t1, end_loop

    # Charger T[i]
    lw $t7, 0($t0)      # $t7 = T[i]

    # Vérifier si T[i] == 0
    beq $t7, $zero, zero_case

    # Cas où T[i] != 0
    li $t3, 0           # Réinitialiser la longueur courante
    li $t6, -1          # Réinitialiser la position de début
    j next_element
```

```
    addi $t2, $t2, 1    # Incrémenter i
    addi $t0, $t0, 4    # Avancer l'adresse dans le tableau
    j loop

end_loop:
    # Stocker le résultat (position de la plus longue suite de zéros) dans "result"
    la $t8, result
    sw $t5, 0($t8)


    li $v0, 1
    move $a0, $t5
    syscall

    # Fin du programme
    li $v0, 10          # Code de terminaison
    syscall
```

Computer Science Department **NAME :…… ……. …….**
2nd Year Bachelor's Degree
Module : Computer Architecture **First Name** :……
Duration : 1H30 / Date : january2025 **GROUP** : ……

Final Exam (Answer on the provided sheet)

Comprehension Questions (5 pts) **:** Answer with true **(T)** or false **(F)** for each of the following statements. A correct answer is worth **(+0.5 pt),** an incorrect answer is worth **(-0.5 pt).**

1. An accumulator-based architecture uses multiple registers to store operands	
2. Control hazards are caused by dependent instructions using the same register.	
3. Forwarding is a solution for structural hazards in a pipeline.	
4. A Von Neumann architecture separates buses for instructions and data.	
5. Pipelining allows executing multiple instructions in the same clock cycle.	
6. DRAMs are memories used in the design of cache memory.	
7. The registers in MIPS architecture are organized in main memory.	
8. Using a branch predictor can reduce data hazards in a pipeline.	
9. The width of the address bus has no impact on the total memory capacity.	
10. A multi-cycle processor requires more hardware than a pipelined processor.	

Exercise 1 (5 pts) :

Given the expression: **Z= (A+B/C)*(D+E)**	
Write a program that calculates the expression Z on a one-address machine.	**Write a program that calculates the expression Z on a zero-address machine.**
Architecture ? ……	**Architecture** ? ……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……

Exercise 2 (5 pts) :

1. Write the MIPS assembly code for the following C code (3 pts):	……
While (X!= Y) **{** **İf (X < Y)** **{ Y = Y – X; }** **Else** **{ X = X – Y; }** **}**	……

Notes : – Declare and initialize X and Y in memory. – Use the registers $t1 and $t2 to load the values of X and Y from memory. – Each syntax error results in (-0.25 pt).	…………………………………………………………… …………………………………………………………… …………………………………………………………… …………………………………………………………… …………………………………………………………… …………………………………………………………… ……………………………………………………………

2. **What does this code do? (2 pts)**

………

………

Exercise 3 (5 pts) : MIPS Programming

Let **T** be an array of 10 integers, with **T(i)∈{0,1}.** Write a MIPS program that returns the **position i** in the array such that **T[i]** is the **beginning of the longest consecutive sequence of zeros**.

Example : T : 1 0 0 1 1 0 0 0 1 0 >>>> **Position i = 5**