



Examen Final

Nom:..... **Prénom:**..... **Groupe:**.....

A. Questions de compréhension...[20%].....

B. Exercices d'application [40%].....

C. Programmation MIPS [40%].....

**N
O
T
E**

--

Partie réservée
à l'enseignant

Lire très attentivement les questions, Répondez sur la feuille

Partie A : Questions de compréhension (10 pts)

1. Quels sont les liens physiques entre le processeur et la mémoire ?

--

2. Qu'est-ce qu'un séquenceur ?

--

3. Qu'est-ce que la hiérarchie mémoire ?

--

4. Qu'est-ce que l'adressage indirect ? A quoi sert-il ?

--

5. Qu'est-ce qu'une architecture pipeline ? Quel gain peut-on espérer avec ce type de dispositif ?
Quelles en sont les limites de gain d'une architecture pipeline ?

--

6. Quels sont les deux principes qui permet d'espérer un tel gain (expliquez ce qu'ils signifient) ?

--

7. Quelles sont les méthodes utilisées pour assurer la cohérence des données entre la mémoire cache et la mémoire principale lors d'une opération d'écriture ?

--

8. Que signifie les acronymes suivants ?

CISC :

LRU :

BIOS :

USB :

Partie B : Exercices d'application [Comptabilisé comme une 2^{ème} Micro-interrogation]**Exercice n°1 : (4 pts)**

- 1) Vous disposez d'un cache d'une capacité de 64 Ko. Combien de lignes peut contenir le cache si les longueurs de ligne sont 64 ou 128 octets ?

.....

.....

.....

.....

.....

.....

.....

.....

- 2) Si un cache possède une capacité de 16 Ko et une longueur de ligne de 128 octets. Combien d'ensembles le cache possède-t-il s'il est associatif par ensemble de 4 ou 8 blocs ?

.....

.....

.....

.....

.....

.....

.....

.....

Exercice n°2 : (6 pts)

Supposez le code MIPS suivant:

```
1      lw      $r1, 0($r0)
2      lw      $r2, 0($r1)
3      add     $r6, $r5, $r4
4      add     $r3, $r1, $r2
5      lw      $r4, 0($r6)
6      sub     $r2, $r0, $r4
7      addi    $r7, $r1, 4
8      add     $r4, $r1, $r3
9      sub     $r6, $r7, $r4
```

- 1) Indiquez toutes les dépendances (aléas) présents dans le code (utiliser les n° d'instructions) ?

Aléa de type :.....	Aléa de type :.....	Aléa de type :.....	Aléa de type :.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

- 2) Dressez un schéma montrant l'exécution de ce code dans le temps, cycle d'horloge par cycle d'horloge. Supposez un pipeline à cinq phases (**F**etch, **D**ecode, **E**xecute, **M**emory et **W**riteback) ?

	1	2	3	4	5	6	7	8	9	10	11	12	13
1 lw \$r1, 0(\$r0)	F	D	E	M	W								
2 lw \$r2, 0(\$r1)		F											
3 add \$r6, \$r5, \$r4													
4 add \$r3, \$r1, \$r2													
5 lw \$r4, 0(\$r6)													
6 sub \$r2, \$r0, \$r4													
7 addi \$r7, \$r1, 4													
8 add \$r4, \$r1, \$r3													
9 sub \$r6, \$r7, \$r4													

	14	15	16	17	18	19	20	21	22	23	24	25	26		
1 lw \$r1, 0(\$r0)															
2 lw \$r2, 0(\$r1)															
3 add \$r6, \$r5, \$r4															
4 add \$r3, \$r1, \$r2															
5 lw \$r4, 0(\$r6)															
6 sub \$r2, \$r0, \$r4															
7 addi \$r7, \$r1, 4															
8 add \$r4, \$r1, \$r3															
9 sub \$r6, \$r7, \$r4															

- 3) Calculez le nombre d'instructions par cycle (IPC) de ce code ?

.....

.....

.....

Partie C : Programmation MIPS [Comptabilisé comme une 2^{ème} Note de TP]

Exercice n°1 (3 pts): Donnez l'équivalent en MIPS de la boucle WHILE (uniquement) ci dessous :

```
int t0 = 100;
int t1 = 0;
int t2 = 10000;
```

```
while ( $t0 < $t2 )
{ t2 = t2 - t1;
++t1; }
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Exercice n°2 :

Supposons le segment de données suivant d'un programme MIPS:

Size: .word 10

List: .word 2, 3, 5, 7, 9, 11, 13, 17, 19, 23

- 1) Ecrire une séquence d'instructions MIPS pour transférer le quatrième élément du tableau **List** dans le registre \$s0 (**2 pts**).

.....
.....
.....
.....
.....
.....
.....
.....

- 2) Ecrire une séquence d'instructions MIPS pour remplacer chaque élément du tableau avec son carré. Votre solution doit être générale dans le sens où il ne devrait pas dépendre des valeurs spécifiques indiquées dans le segment de données ci-dessus. Commentez votre code pour expliquer la signification logique de chaque instruction. **(5 pts)**



Correction Examen Final

Partie réservée à l'enseignant	Questions de compréhension... [20%].....	NOTE	
	Exercices d'application [40%].....		
	Programmation MIPS [40%].....		

Partie A : Questions de compréhension (10 pts)

1. Quels sont les liens physiques entre le processeur et la mémoire ?

le bus mémoire est constitué :

du bus d'adresse

du bus de données

du bus de contrôle

2. Qu'est-ce qu'un séquenceur ?

Le séquenceur organise (synchronise) l'exécution des instructions selon le rythme de l'horloge, il génère les signaux nécessaires pour exécuter une instruction.

3. Qu'est-ce que la hiérarchie mémoire ?

L'ensemble des dispositifs de stockage différenciés et organisés par leur capacité et leur débit.

4. Qu'est-ce que l'adressage indirect ? A quoi sert-il ?

L'opérande n'est pas donnée (implicite), mais elle est dans un registre (spécifié par code opération).

Cette valeur est une adresse (comme en direct et non comme implicite ou immédiat) Utilisée lors d'utilisation (parcours) de tableaux (ou de chaînes).

5. Qu'est-ce qu'une architecture pipeline ? Quel gain peut-on espérer avec ce type de dispositif ?

Quelles en sont les limites de gain d'une architecture pipeline ?

Travail à la chaîne.

Gain = x nb postes de travail (unités), ou profondeur ou nombre d'étages du pipeline

Limite = conflits (de données, structurels, de contrôle)

6. Qu'est-ce que la mémoire cache ? Quel gain peut-on espérer avec ce type de dispositif ?

Une mémoire rapide qui met à disposition des données d'une mémoire plus lente. On peut espérer avoir l'illusion de la rapidité de la plus petite tout en conservant la capacité de l'ensemble

7. Quels sont les deux principes qui permettent d'espérer un tel gain (expliquez ce qu'ils signifient) ?

Les principes de localité spatiale et temporelle. Les données utilisées sont souvent les mêmes (répétitions) ou dans le même coin de mémoire (code ou données)

8. Quelles sont les méthodes utilisées pour assurer la cohérence des données entre la mémoire cache et la mémoire principale lors d'une opération d'écriture ?

Write through et write back

9. Que signifient les acronymes suivants ?

CISC : Complex Instruction Set Computer

LRU : Last recently Used

BIOS : Basic Input Output System

USB : Universal Serial Bus

Partie B : Exercices d'application

Exercice n°1 : (4 pts)

- 1) Vous disposez d'un cache d'une capacité de 64 Ko. Combien de lignes peut contenir le cache si les longueurs de ligne sont 64 ou 128 octets ?

Correction : Le nombre de lignes d'un cache correspond simplement à sa capacité divisée par la longueur de ligne, aussi le cache possède-t-il 1024 lignes si les lignes font 64 octets, 512 si elles font 128 octets et 256 si elles en font 128.

- 2) Si un cache possède une capacité de 16 Ko et une longueur de ligne de 128 octets. Combien d'ensembles le cache possède-t-il s'il est associatif par ensemble de 4 ou 8 blocs ?

Correction : Avec des lignes de 128 octets, le cache contient un total de 128 lignes. Le nombre d'ensembles dans le cache correspond au nombre de lignes divisé par le degré d'associativité du cache. Le cache possède donc 64 ensembles pour une associativité par ensembles de 2 blocs, 32 ensembles pour une associativité par ensemble de 4 blocs et 16 ensembles pour une associativité par ensemble de 8 blocs.

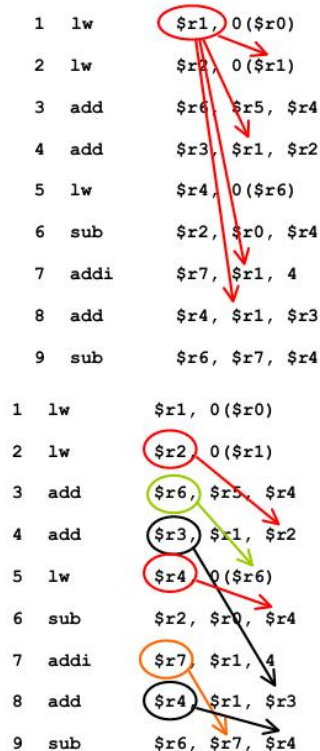
Exercice n°2 : (6 pts)

Supposez le code MIPS suivant:

```
1      lw      $r1, 0($r0)
2      lw      $r2, 0($r1)
3      add     $r6, $r5, $r4
4      add     $r3, $r1, $r2
5      lw      $r4, 0($r6)
6      sub     $r2, $r0, $r4
7      addi    $r7, $r1, 4
8      add     $r4, $r1, $r3
9      sub     $r6, $r7, $r4
```

- 4) Indiquez toutes les dépendances (aléas) présents dans le codes ?

- Dépendances RAW (Read After Write):
 - Après chaque écriture d'un registre, vérifier si le même registre est lu



- **Dépendances WAR (Write After Read):**
 - Après chaque lecture d'un registre, vérifier si le même registre est destination

```

1 lw    $r1, 0($r0)
2 lw    $r2, 0($r1)
3 add   $r6, $r5, $r4
4 add   $r3, $r1, $r2
5 lw    $r4, 0($r6)
6 sub   $r2, $r0, $r4
7 addi  $r7, $r1, 4
8 add   $r4, $r1, $r3
9 sub   $r6, $r7, $r4

```

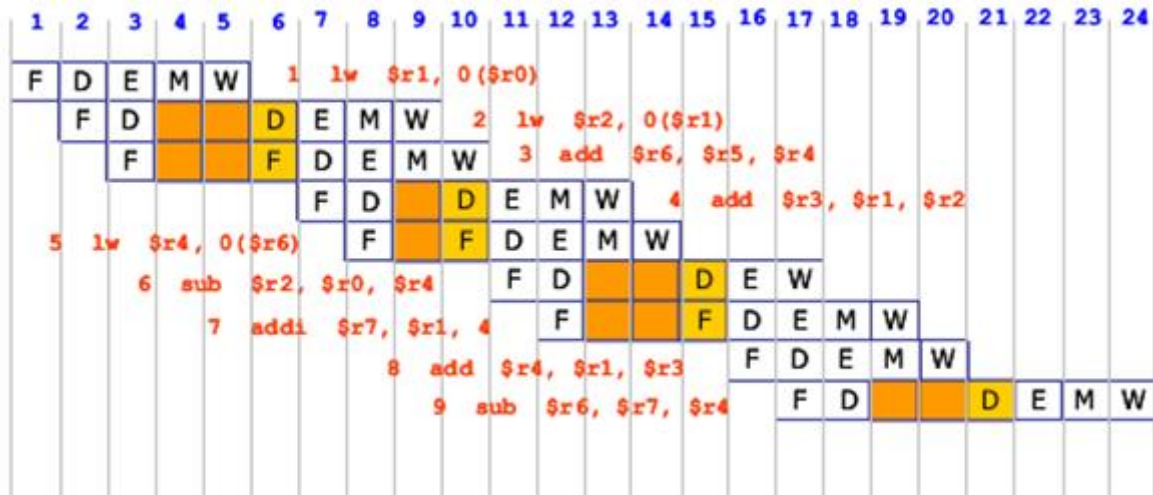
- **Dépendances WAW (Write After Write):**
 - Après chaque écriture d'un registre, vérifier si le même registre est destination

```

1 lw    $r1, 0($r0)
2 lw    $r2, 0($r1)
3 add   $r6, $r5, $r4
4 add   $r3, $r1, $r2
5 lw    $r4, 0($r6)
6 sub   $r2, $r0, $r4
7 addi  $r7, $r1, 4
8 add   $r4, $r1, $r3
9 sub   $r6, $r7, $r4

```

- 5) Dressez un schéma montrant l'exécution de ce code dans le temps, cycle d'horloge par cycle d'horloge. Supposez un pipeline à cinq phases (Fetch, Decode, Execute, Memory et Writeback) ?



- 6) Calculez le nombre d'instructions par cycle (IPC) de ce code ?

$$IPC = (9 \text{ instructions}) / (24 \text{ cycles}) = 0.375$$

Partie C : Programmation MIPS

Exercice n°1(3 pts) : Donnez l'équivalent en MIPS de la boucle WHILE ci dessous :

```
        bge $t0, $t2, ldone
loop:   sub $t2, $t2, $t1      # t2 = t2 - t1
        addi $t1, $t1, 1      # ++t1
        blt $t0, $t2, loop
ldone:
```

Exercice n°2 :

Supposons le segment de données suivant d'un programme MIPS:

Size: .word 10

List: .word 2, 3, 5, 7, 9, 11, 13, 17, 19, 23

- 1) Ecrire une séquence d'instructions MIPS pour transférer le quatrième élément du tableau **List** dans le registre \$s0 **(2 pts)**.

Sol 1:

```
lw $s0, List + 12      # note that the fourth word would be 12 bytes
                        # from the beginning of the array
```

Sol 2:

```
la $t0, List
lw $s0, 12($t0)
```

- 2) Ecrire une séquence d'instructions MIPS pour remplacer chaque élément du tableau avec son carré. Votre solution doit être générale dans le sens où il ne devrait pas dépendre des valeurs spécifiques indiquées dans le segment de données ci-dessus. Commentez votre code pour expliquer la signification logique de chaque instruction **(5 pts)**.

```
.data
Size: .word 10
List: .word 2, 3, 5, 7, 9, 11, 13, 17, 19, 23
.text
la $t0, List          # get a pointer into the array
lw $t1, Size          # get upper limit for loop iterations
li $t2, 0             # initialize loop counter
loop: bge $t2, $t1, exit # check iteration count
lw $s0, ($t0)         # load current value from list
mul $s0, $s0, $s0     # square current value
sw $s0, ($t0)         # write square back to list
addi $t0, $t0, 4      # move pointer to next element
addi $t2, $t2, 1      # increment loop counter
b loop               # restart loop
exit: li $v0, 10      # exit program
syscall
```