ATTENTION : Faire les parties 1, 2 et 3 sur des **feuilles séparées**

1 Hiérarchie mémoire et transformation affine d'images

La transformation affine d'image est à la base de nombreux algorithmes de traitement d'image. Pour transformer une image, la transformation inverse est appliquée sur les coordonnées de l'image résultat: un pixel p' de l'image résultat prend la valeur du pixel p de l'image originale, avec $p = Ap'$, avec A la matrice

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \quad p' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \text{ et } \quad p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

L'algorithme de transformation consiste à parcourir tous les pixels p' de l'image résultat et à leur affecter la valeur du pixel p correspondant.

Les pixels d'une image sont stockés dans un tableau 2D dont le rangement en mémoire se fait selon l'ordre cannonique. C'est à dire qu'un pixel de coordonnée (x, y) est stocké à l'adresse $@ = t_x * y + x$, avec t_x la taille en x de l'image.

1.1 Optimisation des accès mémoire

On considère des images originale et résultat de taille $(512, 512)$, un pixel nécessite 8 bits. On s'intéresse à l'optimisation des accès mémoire lors d'une simple rotation. Dans ce cas, la matrice de transformation vaut:

$$A = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Pour simplifier les équations, l'origine des axes est prise au centre de l'image.

Le code de transformation est le suivant:

```
for x=0 to 511
 for y=0 to 511
  resultat[x,y]=image[a*x+b*y][d*x+e*y]
```

Ce code est executé sur processeur DSP de type TM32, avec un cache dont les caractéristiques sont les suivantes:

Taille	16 KByte
Ligne	64 Byte
Set-associativity	8-way

▷ **Question 1 :**

- Quel est le nombre de lignes de cache ?
- Quel est le nombre de groupes ?

▷ **Question 2 :**

- Quels sont les bits d'adresse utilisés pour calculer le N° du groupe ?
- Quelle est l'écart minimum des adresses de deux données situées dans deux lignes différentes et dans le même groupe ?

▷ **Question 3 :**
Lorsque y vaut 0 (première ligne), combien de lignes de cache sont elles nécessaires lorsque x varie de 0 à 511 dans les trois cas suivants:

- $\theta = 0$ (simple recopie)
- $\theta = \frac{\pi}{2}$ (rotation 90°)
- $\theta = \frac{\pi}{4}$ (rotation 45°)

▷ **Question 4 :**
Pour $x = 1$, dans chacun des cas, est-il possible de réutiliser les lignes précedement utilisées ? Quel est le nombre de miss de cache ?

Pour résoudre ce problème, une solution est de modifier le code de la façon suivante:

```
for xtuile=0 to 511 step 16
 for ytuile=0 to 511 step 16
  for dx=0 to 15
   for dy=0 to 15
    x=xtuile+dx
    y=ytuile+dy
    resultat[x,y]=image[a*x+b*y][d*x+e*y]
```

▷ **Question 5 :**
Pour une rotation, de combien de pixels a-t-on besoin pour calculer une tuile (i.e `xtuile` et `ytuile` constant) ?

▷ **Question 6 :**
Lorsque `xtuile` et `ytuile` sont constants, dans les trois cas suivants:

- $\theta = 0$ (simple recopie)
- $\theta = \frac{\pi}{2}$ (rotation 90°)
- $\theta = \frac{\pi}{4}$ (rotation 45°)

- combien de lignes de cache sont-elles nécessaires pour calculer une tuile ?
- Combien de lignes sont-elles dans le même groupe ?
- Quel est le nombre de miss de cache dans les 3 cas ?

▷ **Question 7 :**
Quelle est l'influence de la taille de la tuile sur le nombre de défauts de cache global ?

1.2 Questions subsidiaires

1.2.1 Algorithmique

▷ **Question 8 :**
Pourquoi faire la transformation dans ce sens plutôt que de remplir l'image résultat en calculant $p' = A^{-1}p$, en parcourant l'image originale ?

1.2.2 Gestion de la précision

Pour un angle de rotation quelconque, il est évident que les coefficients de la matrice sont des réels dans l'intervalle $[-1, 1]$. Afin de produire des images de bonne qualité, il est nécessaire de procéder à une interpolation bi-linéaire des pixels de l'image originale. Le cahier des charges fixe comme contrainte que les coordonnées p doivent être au quart de pixel près.

▷ **Question 9 :**
Donner la précision (partie entière et partie fractionnaire) des coefficients de la matrice A, en expliquant la méthode de calcul.

2 Déroulement de boucle et ordonnancement d'instructions

Le produit scalaire de 2 vecteurs $\vec{x} = (x_1, \ldots, x_n)$ et $\vec{y} = (y_1, \ldots, y_n)$ dans un espace de dimension n se calcule comme $\sum_{i=1}^{n} x_i \cdot y_i$. Traduit en assembleur MIPS, cela donne (avec F2 initialement à zéro et R1 pointant sur x_1 et R2 sur y_1, sachant que les $\vec{x}$ et $\vec{y}$ sont en fait des tableaux de flottant double précision - 8 octets -, et R3 contient initialement n) :

```
loop: L.D F0, 0(R1)     % charge x[i]
      L.D F4, 0(R2)     % charge y[i]
      MUL.D F0, F0, F4  % calcul de x[i] * y[i]
      ADD.D F2, F2, F0  % somme à l'existant
      ADDUI R1, R1,  8  % passe à l'x suivant
      ADDUI R2, R2,  8  % passe à l'y suivant
      ADDUI R3, R3, -1  % décrémente le compteur
      BNEZ  R3, loop    % itère si pas au bout
      S.D   F2          % sauve le resultat
```

On a les gels (*stall*) suivants (identiques à ceux pris en cours) :

Instruction produisante	Instruction consommante	Gels (cycles)
FP ALU op	FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
FP op	INT ALU op	0
INT ALU op	INT ALU op	0
INT ALU op	branch	1

▷ **Question 10 :**
Réécrivez la boucle telle quelle en faisant apparaître explicitement les cycles de gels dues au dépendances et anti-dépendances de données.

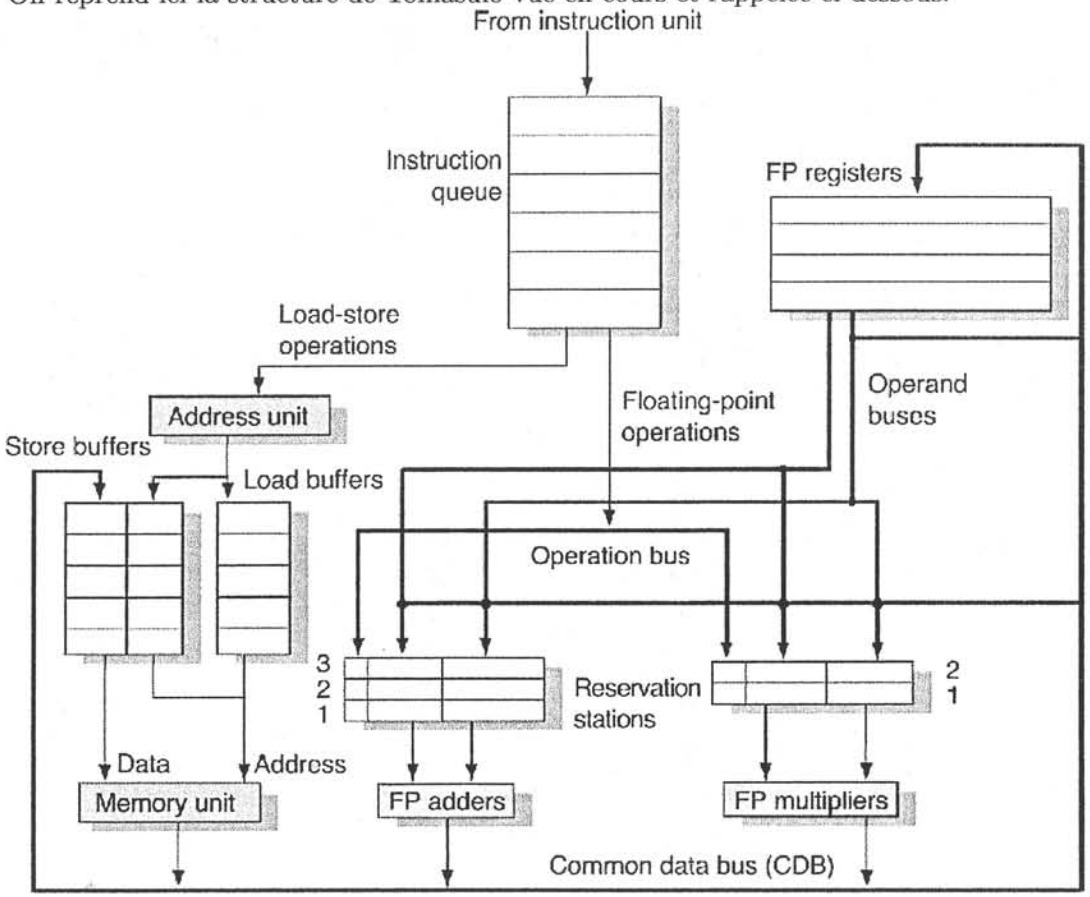
▷ **Question 11 :**
Réordonnancez les instructions de sorte à minimiser ce nombre de cycles.

▷ **Question 12 :**
S'il reste des cycles, déroulez la boucle du nombre de cycles nécessaires pour les faire disparaître.

▷ **Question 13 :**
En supposant que l'on ait une unité entière et une unité flottante qui utilisent des ressources totalement différentes (hormis bien sur les registres entiers), réécrivez la boucle. On utilisera 2 colonnes, l'une pour les instructions flottantes, l'autre pour les instructions entières.

3 Algorithme de Tomasulo

On reprend ici la structure de Tomasulo vue en cours et rappelée ci-dessous.



Et on fait l'hypothèse des latences suivantes pour les opérations flottantes (elle est de 0 pour les opérations entières).

opération	latence
load	2 cycles
add/sub	2 cycles
multiply	10 cycles

▷ **Question 14 :**
Exécutez deux itérations de la partie flottante de la boucle de l'exercice précédent sur la structure de Tomasulo en détaillant pour chaque instruction la station de réservation dans laquelle elle patiente ainsi que son état, afin de mettre en évidence les dépendances.

On peut utiliser un ou plusieurs gabarit pour expliquer l'analyse, qu'il faut rendre en fin d'examen en ce cas.

Instruction status

Instruction	j	k	Issue	Execution complete	Write Result
L.D F0, 0(R1)					
L.D F4, 0(R2)					
MUL.D F0, F0, F4					
ADD.D F2, F2, F0					
ADDUI R1, R1, 8					
ADDUI R2, R2, 8					
ADDUI R3, R3, -1					
BNEZ R3, loop					
S.D F2					

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								

Instruction status

Instruction	j	k	Issue	Execution complete	Write Result
L.D F0, 0(R1)					
L.D F4, 0(R2)					
MUL.D F0, F0, F4					
ADD.D F2, F2, F0					
ADDUI R1, R1, 8					
ADDUI R2, R2, 8					
ADDUI R3, R3, -1					
BNEZ R3, loop					
S.D F2					

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								

Instruction status

Instruction	j	k	Issue	Execution complete	Write Result
L.D F0, 0(R1)					
L.D F4, 0(R2)					
MUL.D F0, F0, F4					
ADD.D F2, F2, F0					
ADDUI R1, R1, 8					
ADDUI R2, R2, 8					
ADDUI R3, R3, -1					
BNEZ R3, loop					
S.D F2					

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								

Instruction status

Instruction	j	k	Issue	Execution complete	Write Result
L.D F0, 0(R1)					
L.D F4, 0(R2)					
MUL.D F0, F0, F4					
ADD.D F2, F2, F0					
ADDUI R1, R1, 8					
ADDUI R2, R2, 8					
ADDUI R3, R3, -1					
BNEZ R3, loop					
S.D F2					

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	No					
0	Add2	No					
	Add3	No					
0	Mult1	No					
0	Mult2	No					

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								