TD-1 : Exécution de boucle avec processeurs scalaires/superscalaire statique : déroulage de boucles, SIMD

Exécution de boucles

Soit un processeur à ordonnancement statique qui a les caractéristiques suivantes :
- les instructions sont de longueur fixe (32 bits)
- Il a 32 registres entiers (dont R0=0) de 32 bits et 32 registres flottants (de F0 à F31) de 32 bits.
- Il peut lire et exécuter 4 instructions par cycle.
- L'unité entière contient deux pipelines d'exécution entière sur 32 bits, soit deux additionneurs, deux décaleurs. Tous les bypass possibles sont implantés.
- L'unité flottante contient un pipeline flottant pour l'addition et un pipeline flottant pour la multiplication. - L'unité Load/Store peut exécuter jusqu'à deux chargements par cycle, mais ne peut effectuer qu'un load et un store simultanément. Elle ne peut effectuer qu'un seul store par cycle.
- Il dispose d'un mécanisme de prédiction de branchement qui permet de "brancher" en 1 cycle si la prédiction est correcte. Les sauts et branchements ne sont pas retardés.

La Table 1 donne
- les instructions disponibles
- le pipeline qu'elles utilisent : E0 et E1 sont les deux pipelines entiers, FA est le pipeline flottant de l'addition et FM le pipeline flottant de la multiplication. Les instructions peuvent être exécutées simultanément si elles utilisent chacune un pipeline séparé.

L'ordonnancement est statique. Les chargements ne peuvent pas passer devant les rangements en attente.

JEU D'INSTRUCTIONS (extrait)

LF	LF Fi, dép.(Ra)	E0 ou E1	Fi <- M (Ra + dépl.16 bits avec ES)
SF	SF Fi, dép.(Ra)	E0	Fi -> M (Ra + dépl.16 bits avec ES)
ADD	ADD Rd,Ra, Rb	E0 ou E1	Rd <- Ra + Rb
ADDI	ADDI Rd, Ra, IMM	E0 ou E1	Rd <- Ra + IMM-16 bits avec ES
SUB	SUB Rd,Ra, Rb	E0 ou E1	Rd <- Ra - Rb
FADD	FADD Fd, Fa, Fb	FA	Fd <- Fa + Fb
FMUL	FMUL Fd, Fa, Fb	FM	Fd <- Fa x Fb
BEQ	BEQ Ri, dépl	E1	si Ri=0 alors CP <- CP+4 + depl
BNE	BNE Ri, dépl	E1	si Ri≠0 alors CP <- CP+4 + depl

Table 1 : instructions disponibles

La Table 2 donne la latence entre une instruction source et une instruction destination, dans le cas de dépendances de données. La valeur 1 est le cas où les deux instructions peuvent se succéder normalement, d'un cycle i au cycle i+1.

Latences *SOURCE* / *DESTINATION*		UAL	LF/SF (données)	Opérations flottantes
UAL		1	2	
LF/ST (adresses)		1	3	
LF/SF (données)		1	2	4
Opérations flottantes			2	4

Table 2 : latences

Soit le programme C suivant (SAXPY) où x et y sont des vecteurs et a est un scalaire de nombres flottants simple précision.

```
float x[1000], y[1000], a;
main ()
{
int i ;
for (i=0; i<1000 ;i++)
                    y[i] = y[i] + a*x[i] ;
}
```

On utilisera les registres suivants : R1 pointe sur x[i], R2 pointe sur y[i]. R3 a été initialisé à 1000. F0 contient a. F1 et F2 recevront respectivement x[i] et y[i].

Question 1

Donner l'exécution cycle par cycle de la boucle optimisée en considérant une version scalaire du processeur et des caches parfaits. Quel est le nombre de cycles par itération de la boucle initiale ?
Donner la version déroulée (4 itérations par corps de boucle) avec la version scalaire en supposant des caches parfaits. Quel est le nombre de cycles par itération de la boucle initiale ?

Question 2

Donner l'exécution cycle par cycle de la boucle optimisée pour la version superscalaire en considérant des caches parfaits. Quel est le nombre de cycles par itération de la boucle initiale ?

Question 3

Donner la version déroulée (4 itérations par corps de boucle) avec la version superscalaire en supposant des caches parfaits. Quel est le nombre de cycles par itération de la boucle initiale ?

Instructions SIMD

On ajoute au processeur 8 registres de 128 bits S0 à S7 pouvant contenir chacun 4 flottants simple précision et les instructions SIMD données dans la Table 3. Cette table donne également les latences des instructions SIMD et le pipeline utilisé dans le cas superscalaire.

PLF	PLF Si, dép.(Ra)	2	E0	Charge quatre flottants simple précision à partir de l'adresse (Ra + dépl.16 bits avec ES).
PST	PST Si, dép.(Ra)	2	E0	Range en mémoire à partir de l'adresse (Ra + dépl.16 bits avec ES) quatre flottants simple précision
PADD	PADD Sf,Sa, Sb	4	FA	SF <- Sa + Sb
PSUB	PSUB Sf ,Sa, Sb	4	FA	SF <- Sa - Sb
PMUL	PMUL Sf, Sa, Sb	4	FM	SF <- Sa x Sb
PMULS	PMULS Sf, Sa, Fb	4	FM	SF <- Sa x Fb (chaque élément de Sa est multiplié par Fb et rangé dans l'élément correspondant de SF)

Table 3 : Instructions SIMD

Question 4

Pour une version scalaire du processeur, en supposant des caches parfaits, donner l'exécution cycle par cycle de la boucle SAXPY en utilisant des instructions SIMD. Quel est le temps d'exécution par itération de la boucle initiale ?

Question 5

Pour la version superscalaire du processeur, en supposant des caches parfaits, donner l'exécution cycle par cycle de la boucle SAXPY en utilisant des instructions SIMD. Quel est le temps d'exécution par itération de la boucle initiale ?

Question 6

Pour la version superscalaire du processeur, en supposant des caches parfaits, donner l'exécution cycle par cycle de la boucle SAXPY en utilisant des instructions SIMD et un déroulage d'ordre 4 (16 itérations de la boucle initiale par corps de boucle). Quel est le temps d'exécution par itération de la boucle initiale ?

CONVERSION SI

On rajoute au processeur les instructions suivantes :

FCMPyy	FCMPyy Fd, Fa, Fb	FA	4	Fd <-1.0 si Fa yy Fb, Fd <- 0.0 sinon avec yy = EQ,NE, GT, GE, LT, LE
FBxx	FBxx Fi, dépl	FA	-	si Fi xx 0.0 alors CP <- NCP + depl avec xx= EQ,NE, GT, GE, LT, LE NCP est l'adresse de l'instruction après FBxx
Bxx	Bxx Ri, dépl	E1	-	si Ri xx 0 alors CP <- NCP + depl avec xx= EQ,NE, GT, GE, LT, LE NCP est l'adresse de l'instruction après Bxx

FCMOVyy	FCMOVyy Fc, Fa, Fb	FA	4 cycles	si Fc yy 0, alors Fa <- Fb avec yy = EQ,NE, GT, GE, LT, LE

Les branchements conditionnels Bxx Ri, dep (pipe E1) peuvent s'exécuter dans le même cycle que l'instruction fournissant Ri (pipe E0).
Les branchements conditionnels flottants FBxx Fi, dep ne peuvent s'exécuter qu'au cycle i+4 si l'instruction FCMPyy est au cycle i (cette dernière a une latence de 4).

Soit le programme

```
float x[100], y[100], z[100], a;

main ()
{
int i ;
for (i=0;i<100;i++)
      {
      a = x[i]+y[i];
      if (a <0.0)
            a=0.0;
      z[i] = a;
      }
}
```

Question 7

Donner le nombre de cycles par itération de la boucle interne **optimisée** en supposant des branchements parfaitement prédits.

a) Sans utiliser FCMOVyy
b) En utilisant FCMOVyy

Question 8

Refaire la question 5 avec un déroulage de boucle d'ordre 4 (4 itérations de la boucle initiale par corps de la boucle déroulée)

c) Sans utiliser FCMOVyy
d) En utilisant FCMOVyy

Question 9

Quelles instructions SIMD faut il rajouter pour utiliser la conversion SI avec des instructions SIMD ?