

Corrigé TD 7 : Opérations multicycles

Exercice 1

Un processeur P1 dispose du pipeline à 5 étages pour les opérations entières et les chargements/rangements flottants et des pipelines suivants pour les instructions flottantes

Multiplication : FMUL

LI	DI	LR	EX1	EX2	EX3	EX4	ER
----	----	----	-----	-----	-----	-----	----

Addition : FADD

LI	DI	LR	EX1	EX2	ER
----	----	----	-----	-----	----

Les instructions flottantes LF et ST ont les mêmes pipelines que les instructions entières LW et SW.

Donner les latences des instructions flottantes FMUL et FADD (une instruction i a une latence de n si l'instruction suivante peut commencer au cycle $i+n$; une latence de 1 signifie que l'instruction suivante peut commencer au cycle suivant).

Latence de FMUL : 4 cycles

Latence de FADD : 2 cycles

Mêmes questions avec le processeur P2 suivant

Multiplication : FMUL

LI	DI	LR	EX1	EX2	EX3	EX4	EX5	EX6	ER
----	----	----	-----	-----	-----	-----	-----	-----	----

Addition : FADD

LI	DI	LR	EX1	EX2	EX3	EX4	ER
----	----	----	-----	-----	-----	-----	----

Latence de FMUL : 6 cycles

Latence de FADD : 4 cycles

Exercice 2

On ajoute au MIPS des instructions flottantes simple précision (32 bits) (Figure 2) et 32 registres flottants F0 à F31 (F0 est un registre normal).

Les additions, soustractions et multiplications flottantes sont pipelinées. Une nouvelle instruction peut démarrer à chaque cycle. Les latences sont de 2 cycles pour LF et données par les résultats de l'exercice 1 pour les instructions flottantes de multiplication et d'addition.

La division a une latence de 30 cycles. Elle n'est pas pipelinée : une nouvelle division ne peut commencer que lorsque la division précédente est terminée.

Boucle SAXPY

Soit le programme suivant

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

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

Quels sont les temps d'exécution par itération de la boucle avec le processeur P1 ? Avec le processeur P2 ?

On utilisera les hypothèses suivantes :

a est initialement dans le registre F0

R1 contient l'adresse de x[0] et R2 contient l'adresse de y[0].

R3 contient l'adresse de x[100], c'est-à-dire l'adresse du mot suivant le dernier élément du tableau x

P1		P2	
cycle	Instruction	cycle	Instruction
1 Boucle	LF F1, 0(R1)	1 Boucle	LF F1, 0(R1)
2	LF F2, 0(R2)	2	LF F2, 0(R2)
3	FMUL F1,F1,F0	3	FMUL F1,F1,F0
4	ADDI R1,R1,4	4	ADDI R1,R1,4
5	ADDI R2,R2,4	5	ADDI R2,R2,4
6		6	
7	FADD F1, F1,F2	7	
8		8	
9	SF F1,-4(R2)	9	FADD F1,F1,F2
10	BNE R1,R3,	10	
		11	
		12	
		13	SF F1,-4(R2)
		14	BNE R1,R3,

10 cycles pour P1 ; 14 cycles pour P2

Soient des déroulages de boucle d'ordre 2 et 4

```
main(){
int i ;
for (i=0 ; i<100 ; i+=2){
    y[i]+= a*x[i];
    y[i+1]+= a*x[i+1];}
}
```

```
main(){
int i ;
for (i=0 ; i<100 ; i+=4){
    y[i]+= a*x[i];
    y[i+1]+= a*x[i+1];
    y[i+2]+= a*x[i+2];
    y[i+3]+= a*x[i+3];}
}
```

Quels sont les temps d'exécution par itération de la boucle initiale pour les 4 cas

- déroulage d'ordre 2 avec P1
- déroulage d'ordre 4 avec P1
- déroulage d'ordre 2 avec P2
- déroulage d'ordre 4 avec P2

Déroulage par deux

P1		P2	
cycle	Instruction	cycle	Instruction
1 Boucle	LF F1, 0(R1)	1 Boucle	LF F1, 0(R1)
2	LF F3, 4(R1)	2	LF F3, 4(R1)
3	LF F2, 0(R2)	3	FMUL F1,F1,F0
4	LF F4, 4(R2)	4	FMUL F3,F3,F0
5	FMUL F1,F1,F0	5	LF F2, 0(R2)
6	FMUL F3,F3,F0	6	LF F4, 4(R2)
7	ADDI R1,R1,8	7	ADDI R1,R1,8
8	ADDI R2,R2,8	8	ADDI R2,R2,8
9	FADD F1, F1,F2	9	FADD F1, F1,F2
10	FADD F3, F3,F4	10	FADD F3, F3,F4
11	SF F1,-8(R2)	11	
12	SF F3,-4(R2)	12	
13	BNE R1,R3,Boucle	13	SF F1,-8(R2)
		14	SF F3,-4(R2)
		15	BNE R1,R3,

Déroulage d'ordre 2

P1 : $13/2 = 6,5$ cycles/itération

P2 : $15/2 = 7,5$ cycles/itération

Déroulage par quatre

P1		P2	
cycle	Instruction	cycle	Instruction
1 Boucle	LF F1, 0(R1)	1 Boucle	LF F1, 0(R1)
2	LF F3, 4(R1)	2	LF F3, 4(R1)
3	LF F5, 8(R1)	3	LF F5, 8(R1)
4	LF F7, 12(R1)	4	LF F7, 12(R1)
5	LF F2, 0(R2)	5	LF F2, 0(R2)
6	LF F4, 4(R2)	6	LF F4, 4(R2)
7	LF F6, 8(R2)	7	LF F6, 8(R2)
8	LF F8, 12(R2)	8	LF F8, 12(R2)
9	FMUL F1,F1,F0	9	FMUL F1,F1,F0
10	FMUL F3,F3,F0	10	FMUL F3,F3,F0
11	FMUL F5,F5,F0	11	FMUL F5,F5,F0
12	FMUL F7,F7,F0	12	FMUL F7,F7,F0

13	FADD F1, F1,F2		13	ADDI R1,R1,16	
14	FADD F3, F3,F4		14	ADDI R2,R2,16	
15	FADD F5, F5,F6		15	FADD F1, F1,F2	
16	FADD F7, F7,F8		16	FADD F3, F3,F4	
17	SF F1,0(R2)		17	FADD F5, F5,F6	
18	SF F3,4(R2)		18	FADD F7, F7,F8	
19	SF F5,8(R2)		19	SF F1,-16(R2)	
20	SF F7,12(R2)		20	SF F3,-12(R2)	
21	ADDI R1,R1,16		21	SF F5,-8(R2)	
22	ADDI R2,R2,16		22	SF F7,-4(R2)	
23	BNE R1,R3,Boucle		23	BNE R1,R3,Boucle	

P1 : $23/4 = 5,75$ cycles/itération
P2 : $23/4 = 5,75$ cycles/itération

Quels sont les temps minimum par itération de la boucle initiale que l'on peut obtenir par déroulage avec chaque processeur ?

On dispose de 32 registres flottants. F0 est utilisé par a. Il faut 2 registres flottants par itérations de la boucle initiale. Le déroulage maximum est donc de 15.

On a alors 30 loads, 15 mul, 15 add, 15 stores, 2 addi et un branchement soit un total de 77 instructions et 77 cycles.

Le temps minimum par itération est donc de $77/15 = 5,13$ cycles (en supposant que le nombre total d'itérations est un multiple de 15)

Comment procède-t-on si le nombre d'itérations n'est pas un multiple du facteur de déroulage ?

Alors on a $\lfloor N/d \rfloor$ itérations avec déroulage et $N\%d$ itérations normales non déroulées.

Somme des carrés

Soit le programme suivant

```
float x[100], s ;

main(){
int i ;
for (i=1 ; i<100 ; i++)
    s+=x[i]*x[i];
}
```

Quels sont les temps d'exécution par itération de la boucle avec le processeur P1 ? Avec le processeur P2 ?

S est initialement dans le registre F0

R1 contient l'adresse de x[0].

R3 contient l'adresse de x[100], c'est-à-dire l'adresse du mot suivant le dernier élément du tableau x

	P1		P2	
cycle	Instruction		cycle	Instruction

	FSUB F0,F0,F0			FSUB F0,F0,F0	
1 Boucle	LF F1, 0(R1)		1 Boucle	LF F1, 0(R1)	
2			2		
3	FMUL F1,F1,F1		3	FMUL F1,F1,F1	
4	ADDI R1,R1,4		4	ADDI R1,R1,4	
5			5		
6			6		
7	FADD F0, F0,F1		7		
8	BNE R1,R3		8		
			9	FADD F0,F0,F1	
			10	BNE R1,R3,	

P1 : 8 cycles/itération
P2 : 10 cycles/itération

Quels sont les temps d'exécution par itération de la boucle initiale pour les 4 cas

- déroulage d'ordre 2 avec P1
- déroulage d'ordre 4 avec P1
- déroulage d'ordre 2 avec P2
- déroulage d'ordre 4 avec P2

Déroulage par 2

	P1			P2	
cycle	Instruction		cycle	Instruction	
	FSUB F0,F0,F0			FSUB F0,F0,F0	
	FSUB F1,F1,F1			FSUB F1,F1,F1	
1 Boucle	LF F2, 0(R1)		1 Boucle	LF F2, 0(R1)	
2	LF F3, 4(R1)		2	LF F3, 4(R1)	
3	FMUL F2,F2,F2		3	FMUL F2,F2,F2	
4	FMUL F3,F3,F3		4	FMUL F3,F3,F3	
5	ADDI R1,R1,8		5	ADDI R1,R1,8	
6			6		
7	FADD F0, F0,F2		7		
8	FADD F1, F1,F3		8		
9	BNE R1,R3, Boucle		9	FADD F0, F0,F2	
E1	FADD F0,F0,F1		10	FADD F1, F1,F3	
E2			11	BNE R1,R3, Boucle	
			E1	FADD F0,F0,F1	

P1 : $9/2 = 4,5$ cycles/itération + 2 cycles pour la somme finale
P2 : $11/2 = 5,5$ cycles/itération + 4 cycles pour la somme finale

Déroulage par 2

	P1			P2	
cycle	Instruction		cycle	Instruction	
	FSUB F0,F0,F0			FSUB F0,F0,F0	

	FSUB F1,F1,F1			FSUB F1,F1,F1	
	FSUB F2,F2,F2			FSUB F2,F2,F2	
	FSUB F3,F3,F3			FSUB F3,F3,F3	
1 Boucle	LF F4, 0(R1)		1 Boucle	LF F4, 0(R1)	
2	LF F5, 4(R1)		2	LF F5, 4(R1)	
3	LF F6, 8(R1)		3	LF F6, 8(R1)	
4	LF F7, 12(R1)		4	LF F7, 12(R1)	
5	FMUL F4,F4,F4		5	FMUL F4,F4,F4	
6	FMUL F5,F5,F5		6	FMUL F5,F5,F5	
7	FMUL F6,F6,F6		7	FMUL F6,F6,F6	
8	FMUL F7,F7,F7		8	FMUL F7,F7,F7	
9	FADD F0,F0, F4		9	ADDI R1,R1,16	
10	FADD F1,F1,F5		10		
11	FADD F2,F2,F6		11	FADD F0,F0, F4	
12	FADD F3,F3,F7		12	FADD F1,F1,F5	
13	ADDI R1,R1,16		13	FADD F2,F2,F6	
14	BNE R1,R3, Boucle		14	FADD F3,F3,F7	
E1	FADD F0,F0,F1		15	BNE R1,R3, Boucle	
E2	FADD F2,F2,F3		E1	FADD F0,F0,F1	
E3			E2	FADD F2,F2,F3	
E4	FADD F0,F0,F2		E3		
E5			E4		
E6	F0 disponible		E5		
			E6	FADD F0,F0,F2	
			E7		
			E8		
			E9		
			E10	F0 disponible	

P1 : $14/4 = 3,5$ cycles par itération + 5 cycles pour somme finale

P2 : $15/4 = 3,75$ cycles par itération + 9 cycles pour somme finale

Quels sont les temps minimum par itération de la boucle initiale que l'on peut obtenir par déroulage avec chaque processeur ?

On dispose de 32 registres flottants

Il faut 2 registres par facteur de déroulage (1 pour la somme partielle, et 1 pour l'accès pour $X[i]$)

Au maximum, on peut dérouler par 16

Il y a alors 16 loads, 16 multiplications, 16 additions, 1 addi et 1 Branchement soit 50 instructions pour 16 itérations.

Soit $50 \text{ cycles}/16 = 3,125$ cycles/itérations auxquels il faut ajouter la sommation finale des 16 sommes partielles (sommation logarithmiques en 4 étapes).

ADD	R	ADD rd, rs, rt	$rd \leftarrow rs + rt$ (signé)
ADDI	I	ADDI rt, rs, IMM	$rt \leftarrow rs + \text{SIMM}$ (signé)
ADDIU	I	ADDIU rt, rs, IMM	$rt \leftarrow rs + \text{SIMM}$ (le contenu des registres est non signé)
ADDU	R	ADDU rd, rs, rt	$rd \leftarrow rs + rt$ (le contenu des registres est non signé)
AND	R	AND rd, rs, rt	$rd \leftarrow rs \text{ and } rt$
ANDI	I	ANDI rt, rs, IMM	$rt \leftarrow rs \text{ and } \text{ZIMM}$
BEQ	I	BEQ rs,rt, déplac.	si $rs = rt$, branche à ADBRANCH
BGEZ	I	BGEZ rs,déplac.	si $rs \geq 0$, branche à ADBRANH
BGEZAL	I	BGEZAL rs, déplac.	adresse de l'instruction suivante dans R31 si $rs \geq 0$, branche à ADBRANH
BGTZ	I	BGTZ rs,déplac.	si $rs > 0$, branche à ADBRANH
BLEZ	I	BLEZ rs,déplac.	si $rs \leq 0$, branche à ADBRANH
BLTZ	I	BLTZ rs,déplac.	si $rs < 0$, branche à ADBRANH
BLTZAL	I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31 si $rs < 0$, branche à ADBRANH
BNEQ	I	BNEQ rs,rt, déplac.	si $rs \neq rt$, branche à ADBRANCH
J	J	J destination	Décale l'adresse destination de 2 bits à gauche, concatène aux 4 bits de poids fort de CP et saute à l'adresse obtenue
JAL	J	JAL destination	Même action que J . Range adresse instruction suivante dans R31
JALR	R	JALR rs, rd	Saute à l'adresse dans rs. Range adresse instruction suivante dans rd
JR	R	JR rs	Saute à l'adresse dans rs
LUI	I	LUI rt, IMM	Place IMM dans les 16 bits de poids fort de rt. Met 0 dans les 16 bits de poids faible de rt
LW	I	LW rt, déplac.(rs)	$rt \leftarrow \text{MEM}[rs + \text{SIMM}]$
OR	R	AND rd, rs, rt	$rd \leftarrow rs \text{ or } rt$
ORI	I	ANDI rt, rs, IMM	$rt \leftarrow rs \text{ or } \text{ZIMM}$
SLL	R	SLL rd, rt, nb	Décale rt à gauche de nb bits et range dans rd
SLT	R	SLT rd, rs, rt	$rd \leftarrow 1$ si $rs < rt$ avec rs signé et 0 autrement
SLTI	I	SLTI rt, rs, IMM	$rt \leftarrow 1$ si $rs < \text{SIMM}$ avec rs signé et 0 autrement
SLTIU	I	SLTIU rt, rs, IMM	$rt \leftarrow 1$ si $rs < \text{ZIMM}$ avec rs non signé et 0 autrement
SLTU	R	SLTU rt, rs, r	$rd \leftarrow 1$ si $rs < rt$ avec rs et rt non signés et 0 autrement
SRA	R	SRA rd, rt, nb	Décaler (arithmétique) rt à droite de nb bits et ranger dans rd
SRL	R	SRL rd, rt, nb	Décaler (logique) rt à droite de nb bits et ranger dans rd.
SUB	R	SUB rd, rs, rt	$rd \leftarrow rs - rt$ (signé)
SUBU	R	SUBU rd rs, rt	$rd \leftarrow rs - rt$ (non signé)
SW	I	SW rt, déplac.(rs)	$rt \Rightarrow \text{MEM}[rs + \text{IMM}]$
XOR	R	XOR rd, rs, rt	$rd \leftarrow rs \text{ xor } rt$
XORI	I	XORI rt, rs, IMM	$rt \leftarrow rs \text{ xor } \text{ZIMM}$

Figure 1 : Instructions entières MIPS utilisées (NB : les branchements ne sont pas retardés)

LF	I	LF ft, déplac(rs)	$rt \leftarrow \text{MEM}[rs + \text{SIMM}]$
SF	I	SF ft, déplac.(rs)	$ft \rightarrow \text{MEM}[rs + \text{SIMM}]$
FADD	R	FADD fd, fs,ft	$fd \leftarrow fs + ft$ (addition flottante simple précision)
FMUL	R	FMUL fd, fs,ft	$fd \leftarrow fs * ft$ (multiplication flottante simple précision)
FSUB	R	FSUB fd, fs,ft	$fd \leftarrow fs - ft$ (soustraction flottante simple précision)
FDIV	R	FDIV fd,fs,ft	$fd \leftarrow fs / ft$ (division flottante simple précision)

Figure 2 : Instructions flottantes ajoutées pour le TD (Ce ne sont pas les instructions MIPS)