

- i) $d * d \geq 0.0$
j) $(d + f) - d == f$

3. Optimisation de programmes flottants.

Le processeur utilisé a le jeu d'instructions MIPS (avec branchements non retardés) donné en annexe. Tous les branchements sont parfaitement prédits (s'exécutent en 1 cycle)

La latence des instructions est définie de la manière suivante : 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.

Les instructions flottantes ont les latences suivantes :

Instructions	Latence	Pipelinée ?
LF	2	OUI
FADD, FSUB	3	OUI
FMUL	5	OUI
FDIV	15	NON

Soient les programmes P1, P2 et P3 :

P1	P2	P3
float X[N], Y[N], Z[N] ; int i ; for (i=0 ; i<N ; i++) Z[i]= X[i] + Y[i] ;	float X[N], Y[N], Z[N], a ; int i ; for (i=0 ; i<N ; i++) Z[i]= X[i] /a + Y[i] ;	int i ; float X[N], Y[N], Z[N], a, c ; c =1.0/a ; for (i=0 ; i<N ; i++) Z[i]= X[i] *c + Y[i] ;

Q 5) Ecrire la version optimisée sans déroulage de boucle du programme P1. Quel est le nombre de cycles par itération et le nombre de cycles total du programme si N=100 ?

Q 6) Ecrire la version optimisée du programme P1 avec un déroulage de boucle d'ordre 2. Quel est le nombre de cycles par itération et le nombre de cycles total si N=100 ?

Q 7) Quelle est la version la plus rapide entre le programme P2 et P3 (justifier). Pour la version la plus rapide,

- écrire la version optimisée sans déroulage de boucle, donner le nombre de cycles par itération et le nombre de cycles total si N=100
- écrire la version optimisée avec déroulage de boucle d'ordre 2, le nombre de cycles par itération et le nombre de cycles total si N=100.

4. Annexe :

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 ADBRANCH
BGEZAL	I	BGEZAL rs, déplac.	adresse de l'instruction suivante dans R31 si $rs \geq 0$, branche à ADBRANCH
BGTZ	I	BGTZ rs,déplac.	si $rs > 0$, branche à ADBRANCH
BLEZ	I	BLEZ rs,déplac.	si $rs \leq 0$, branche à ADBRANCH
BLTZ	I	BLTZ rs,déplac.	si $rs < 0$, branche à ADBRANCH
BLTZAL	I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31. si $rs < 0$, branche à ADBRANCH
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, rt	$rd \leftarrow 1$ 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)