

Corrigé Examen Novembre 04 - Architectures Avancées

3H – Tous documents autorisés

PIPELINES

Soit un processeur qui a les pipelines suivants

Entiers

LI DI LR EX AC LF AT RT ER

Flottants

LI DI LR LRF EX1 EX2 EX3 RT ER

avec la signification suivante :

LI : lecture des instructions dans le cache instructions

DI : décodage des instructions

LR : lecture registres entiers

LRF : lecture des registres flottants

EX : exécution UAL pour les entiers, et calcul des adresses (mémoire et branchements)

EX_i : phase d'une exécution flottante

AC : accès au cache, résolution des branchements (condition connue)

LF : fin de l'accès cache (succès ou échec). Si succès, on dispose à la fin de LF de la donnée lue.

AT : Attente

RT : Résolution des exceptions

ER : Écriture registres.

Tous les "bypass" nécessaires existent.

Question 1

a) Donner les latences entre instruction producteur et instruction consommateur en nombre de cycles.

NB : la valeur n signifie que deux instructions dépendantes peuvent se suivre aux cycles i et i+n, que ce soit pour un processeur scalaire ou superscalaire. Une valeur 0 signifie que les deux instructions peuvent démarrer au même cycle (pour un superscalaire)

	Instruction producteur	Instruction consommateur	Latence superscalaire	Latence scalaire
a	UAL Ri, -, -	UAL -, Ri, -	1	1
b	UAL Ri, -, -	ST Ri, ()	0	1
c	UAL Ri, -, -	LF Rj, (Ri)	0	1
d	LW Ri, ()	UAL -, Ri, -	3	3
e	UAL Ri, -, -	Bconditionnel Ri, déplac.	0	1
f	FOP Fi, -, -	FOP -, Fi, -	3	3
g	LFF Fi, -, -	FOP -, Fi, -	2	2
h	FOP Fi, -, -	STF Fi, (Rj)	3	3

b) Quelle est la pénalité de branchement conditionnel mal prédit ?

Branchement conditionnel mal prédit ?

4 cycles

CONDITIONS DE VECTORISATION SIMD

Question 2

Pour les quatre boucles suivantes, indiquez si elles peuvent utiliser les instructions SIMD (si elles sont « vectorisables »), éventuellement après transformation à indiquer. Sinon, indiquez pourquoi elles ne sont pas vectorisables.

Boucle 1:

```
float X[N], Y[N], Z[N], A, B;
for (i=0; i<N; i++) {
    X[i]=Y[i]+A;
    Z[i]=X[i]+B; }
```

Vectorise

Boucle 2

```
float X[N], Y[N], Z[N], A, B;
for (i=0; i<N-1; i++) {
    X[i]=Y[i]+A;
    Z[i]=X[i+1]+B; }
```

Ne vectorise pas : dépendance entre itérations

Boucle 3

```
float X[N], Y[N], Z[N], A, B;
for (i=0; i<N-1; i++) {
    X[i+1]=Y[i]+A;
    Z[i]=X[i]+B; }
```

Vectorise

Boucle 4

```
float X[N], Y[N], M[N][N];
for (i=0; i<N; i++)
    for (j=0; j<N; j++)
        Y[j] += X[i]*M[i][j] ;
```

Vectorise

Question 3

Transformer la boucle suivante pour la rendre vectorisable.

```
float A[400], B[400], Z;
*p_a = &A[0] ;
*p_b = &B[0] ;
*p_z = &Z ;
int f ;
for (f=0; f<400; f++)
    *p_z += *p_a++ * *p_b++ ;

for (f=0; f<400; f++)
    z+=A[f]*B[f];
```

CACHES

On considère le produit de deux matrices de flottants simple précision sous la forme i, j, k.

```
for (i=0; i<N; i++)
    ....for (j=0; j<N; j++)
    .....{
    .....s=0;
    .....for (k=0; k<N; k++)
    .....{
    .....s = s + x[i][k]*y[k][j];
```

```

.....}
.....z[i][j]=s;
.....}

```

Le cache données L1 du processeur considéré a 16 Ko

Question 4

Donner la taille maximale de la sous matrice [B][B] pour les données nécessaires des matrices x et y puissent rester dans le cache pendant les deux itérations les plus internes du produit de matrices (technique du blocage). On ne tiendra compte que de la capacité du cache, et non d'éventuels conflits.

Il faut qu'une ligne de x, correspondant à x[i][] et N colonnes de y correspondant à y[][] restent dans la matrice, soit (B+1) fois B éléments de 4 octets.

Soit $4 B^2 + 4B < 16 K$

$B^2 + B < 4 K$

Soit $B < 64$

OPTIMISATION DE BOUCLES (1)

On utilise le processeur superscalaire défini dans l'annexe 1

Soit la boucle suivante :

```

float X[128], Y[128], Z[128] ;
for (i = 0 ; i < 128 ; i++)
    Z[i] = (X[i] + Y[i]) * 0.5 ; /*optimisation par rapport à
                                la division par 2.0*/

```

On supposera que l'adresse de X[0] est initialement dans R1, que l'adresse de Y[0] est initialement dans R2 et que l'adresse de Z[0] est initialement dans R3. R4 contient initialement le nombre d'itérations de la boucle. Lorsqu'on rentre dans la boucle, F0 contient 0.5.

Question 5

Quel est en nombre de cycles, le temps d'exécution par itération de la boucle originale sans et avec utilisation des instructions SIMD. ?

Sans déroulage, sans SIMD

	E0	E1	FA	FM
1 Boucle	LF F1, 0(R1)	LF F2, 0(R2)		
2	ADDI R1,R1,4	ADDI R2,R2,4		
3	ADDI R3,R3,4	ADDI R4,R4,-1	FADD F1,F1,F2	
4				
5				
6				
7				FMUL F1,F1,F0
8				
9				
10				
11				
12				
13	SF F1, -4(R3)	BNE R3, Boucle		

	E0	E1	FA	FM
1 Boucle	PLF S1, 0(R1)	ADDI R4,R4,-4		
2	PLF F2, 0(R2)	ADDI R3,R3,16		
3	ADDI R1,R1,16	ADDI R2,R2,16		
4			PADD S1,S1,S2	
5				
6				
7				
8				PMULS S1,S1,F0
9				
10				
11				
12				
13				
14	PSF F1, -16(R3)	BNE R4, Boucle		

Question 6

Optimiser la boucle précédente sans et avec instructions SIMD.

Donner par itération de la boucle initiale

- le nombre de cycles sans instructions SIMD avec un déroulage d'ordre 4
- le nombre de cycles avec instructions SIMD avec un déroulage d'ordre 4

avec déroulage, sans SIMD

	E0	E1	FA	FM
1 Boucle	LF F1, 0(R1)	LF F2, 0(R2)		
2	LF F3, 0(R1)	LF F4, 0(R2)		
3	LF F5, 0(R1)	LF F6, 0(R2)	FADD F1,F1,F2	
4	LF F7, 0(R1)	LF F8, 0(R2)	FADD F3,F3,F4	
5	ADDI R1,R1,16	ADDI R2,R2,16	FADD F5,F5,F6	
6	ADDI R3,R3,16	ADDI R4,R4,-4	FADD F7,F7,F8	
7				FMUL F1,F1,F0
8				FMUL F3,F3,F0
9				FMUL F5,F5,F0
10				FMUL F7,F7,F0
11				
12				
13	SF F1 -16(R3)			
14	SF F3, -12(R3)			
15	SF F5, -8(R3)			
16	SF F7, -4(R3)	BNE R3, Boucle		

Avec déroulage, SIMD

	E0	E1	FA	FM
1 Boucle	PLF S1, 0(R1)	ADDI R3,R3,64		
2	PLF S2, 0(R2)	ADDI R1,R1,64		
3	PLF S3, -48(R1)	ADDI R2,R2,64		
4	PLF S4, -48(R2)	ADDI R4,R4,-16	PADD S1,S1,22	

5	PLF S5, -32(R1)			
6	PLF S6, -32(R2)		PADD S3,S3,S4	
7	PLF S7, -16(R1)			
8	PLF S0, -16(R2)		PADD S5,S5,S6	PMULS S1,S1,F0
9				
10			PADD S7,S7,S0	PMULS S3,S3,F0
11				
12				PMULS S5,S5,F0
13				
14	PSF S1 -64(R3)			PMULS S7,S7,F0
15				
16	PSF S3 -48(R3)			
17				
18	PSF S5 -32(R3)			
19				
20	PSF S7 -16(R3)	BNE R3, Boucle		

Résumé :

Sans déroulage sans SIMD : 13 cycles

Sans déroulage avec SIMD : $14/4 = 3,5$ cycles

Avec déroulage sans SIMD : $16/4 = 4$ cycles

Avec déroulage avec SIMD : $20/16 = 1,25$ cycles

Instructions SIMD

On ajoute au processeur superscalaire décrit dans l'annexe 1 des instructions SIMD pour travailler sur des octets non signés. Elles sont décrites dans l'annexe 1.

Soit le programme suivant

```
unsigned char X[512], Y[512], A[512];
for (i=0 ; i<512 ; i++)
    A[i][j] = abs (X[i][j] -Y[i][j] ) ;
```

On suppose que les adresses de départ des tableaux X, Y et A sont respectivement dans les registres R1, R2 et R3 du processeur avant d'exécuter la boucle. R4 contient le nombre d'itérations (512) avant d'exécuter la boucle

Question 7 : Donner le code assembleur de la boucle avec des instructions SIMD

	E0	E1	FA	FM
Boucle	PLB S0, 0(R1)	ADDI R4,R4,-16		
	PLB S1, 0(R2)	ADDI R1,R1,16		
		ADDI R2,R2,16		
	PSUBU S2,S0,S1	ADDI R3,R3,16		
	PSUBU S3, S1,S0			
	PMAX S2, S2, S3			
	PSB S2, -16(R3)	BNE R3, Boucle		

Autre version

D'abord calcul des max et des min, puis soustraction max -min

Pipeline logiciel avec IA-64

Soient les deux boucles en pipeline logiciel suivantes :

BOUCLE 1 : (Intervalle inter-itérations : 1)
int Y[200], X[200], Z, i ; // (Intervalle inter-itérations : 1)
for (i=0 ; i<200), i++)
 Y[i] = X[i] + Z

 mov lc=199
 mov ec= ??
 mov pr.rot=1<<16 ;;

L1 : (p16) ld4 r32=[r5],4
(???) add ???=???, r9
(???) st4 [r6]=???, 4
br ctop L1 ;;

Question 8 : remplacer les ??? par les numéros de registres si les latences des instructions IA-64 sont celles indiquées dans l'annexe 2 et les données sont dans le cache L1

 mov lc=199
 mov ec= 3
 mov pr.rot=1<<16 ;;

L1 : (p16) ld4 r32=[r5],4
(p17) add r34= r33, r9
(p18) st4 [r6]= r35, 4
br ctop L1 ;;

0	1	2	3	4	5	6	7	8	9
1	LD4								
2	ADD	LD4							
3	ST4	ADD	LD4						
4		ST4	ADD	LD4					
5			ST4	ADD					
6				ST4					

BOUCLE 2 : (Intervalle inter-itérations : 1)
double Y[200], X[200], Z ;
int i
for (i=0 ; i<200), i++)
 Y[i] = X[i] + Z

 mov lc=199
 mov ec= ??

L1 : mov pr.rot=1<<16 ;;
(p16) ldf f32=[r5],8
(???) fadd ???,???, f9
(???) stf8 [r6],???, 8
br ctop L1

Question 9 : remplacer les ??? par les numéros de registres si les latences des instructions IA-64 sont celles indiquées dans l'annexe 2 et les données dans le cache L2.

0													
1	LDF												
2		LDF											
3			LDF										
4				LDF									
5					LDF								
6						LDF							
7	FADD						LDF						
8		FADD						LDF					
9			FADD						LDF				
10				FADD						LDF			
11	STF				FADD						LDF		
		STF				FADD							
			STF				FADD						
				STF				FADD					
					STF				FADD				
						STF				FADD			
							STF				FADD		
								STF				FADD	
									STF				FADD
										STF			
											STF		
												STF	

BOUCLE 2 : (Intervalle inter-itérations : 1)

double Y[200], X[200], Z ;

int i

for (i=0 ; i<200), i++)

 Y[i] = X[i] + Z

 mov lc=199

 mov ec= 11

L1 : mov pr.rot=1<<16 ;;

 (p16) ldf f32=[r5],8

 (p22) fadd f39, f38, f9

 (p26) stf8 [r6], f43, 8

 br ctop L1

Annexe 1

Soit un processeur superscalaire 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'addition et la multiplication flottante sont pipelinées. La division flottante n'est pas pipelinée (une division ne peut commencer que lorsque la division précédente est terminée).

L'ordonnement 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	$Fi \leftarrow M(Ra + \text{dépl.16 bits avec ES})$
SF	SF Fi, dép.(Ra)	E0	$Fi \rightarrow M(Ra + \text{dépl.16 bits avec ES})$
ADD	ADD Rd,Ra, Rb	E0 ou E1	$Rd \leftarrow Ra + Rb$
ADDI	ADDI Rd, Ra, IMM	E0 ou E1	$Rd \leftarrow Ra + \text{IMM-16 bits avec ES}$
SUB	SUB Rd,Ra, Rb	E0 ou E1	$Rd \leftarrow Ra - Rb$
FADD	FADD Fd, Fa, Fb	FA	$Fd \leftarrow Fa + Fb$
FMUL	FMUL Fd, Fa, Fb	FM	$Fd \leftarrow Fa \times Fb$
FDIV	FDIV Fd, Fa, Fb	FA	$Fd \leftarrow Fa/Fb$
BEQ	BEQ Ri, dépl	E1	si $Ri=0$ alors $CP \leftarrow CP + \text{depl}$
BNE	BNE Ri, dépl	E1	si $Ri \neq 0$ alors $CP \leftarrow CP + \text{depl}$

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	<u>Source</u>	UAL	LF/SF (données)	FADD	FMUL	FDIV	FSQRT
	<u>Destination</u>						
UAL		1	2				
LF/ST (adresses)		1	3				
SF (données)		1	2	4	6	20 (NP)	20 (NP)
Opération flottante			2	4	6	20 (NP)	20 (NP)

Table 2 : latences

Le processeur a également 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).
PSF	PSF 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 Sd,Sa, Sb	4	FA	$Sd \leftarrow Sa + Sb$ sur quatre « floats »
PSUB	PSUB Sd,Sa, Sb	4	FA	$Sd \leftarrow Sa - Sb$ sur quatre « floats »
PMUL	PMUL Sd, Sa, Sb	6	FM	$Sd \leftarrow Sa \times Sb$ sur quatre « floats »
PMULS	PMULS Sd, Sa, Fb	6	FM	$SF \leftarrow Sa \times Fb$ (chaque élément de Sa est multiplié par Fb et rangé dans l'élément correspondant de SF) sur quatre « floats »
PLB	PLB Si, dép.(Ra)	2	E0	Charge seize octets à partir de l'adresse (Ra + dépl.16 bits avec ES).
PSB	PSB Si, dép.(Ra)	2	E0	Range en mémoire à partir de l'adresse (Ra + dépl.16 bits avec ES) seize octets
PADDU	PADDU Sd,Sa, Sb	1	E1	$Sd \leftarrow Sa + Sb$: addition saturée sur 16 octets non signés
PSUBU	PSUBU Sd,Sa, Sb	1	E1	$Sd \leftarrow Sa - Sb$: soustraction saturée sur 16 octets non signés
PMAX	PMAXU Sd,Sa, Sb	1	E0	$Sd \leftarrow Sa \max Sb$: maximum (non signé)
PMIN	PMINU Sd,Sa, Sb	1	E0	$Sd \leftarrow Sa \min Sb$: minimum (non signé)

Table 3 : Instructions SIMD

Annexe 2

Latence des instructions IA-64 disponibles

Consommateur (à droite) Producteur (en bas)	UAL	Adresse load store	Store données	Inst. flottantes	getf	setf
Instructions UAL	1	1	1			1
getf			5			5
setf			6			
Instructions flottantes			4	4	4	
Load entier	N	N+1	N		N	N
Load flottant	M+1		M+1	M+1	M+1	M+1

Table 4 : Latence des instructions Itanium 2

N=1 pour cache L1D, N=5 pour L2, N=12-15 pour L3, N=180-220 pour MP
M = 5 pour L2, M=12-15 pour L3, M=180-220 pour MP