Corrigé Examen Novembre 2013 - Architectures Avancées
3H – Tous documents autorisés

OPTIMISATION DE BOUCLES

On utilise le processeur superscalaire défini dans l'annexe 1
Soit la boucle suivante :

```
float X[1024], Y[512];
for (i=0 ; i<512 ; i++)
      Y[i] = X[i+512] – X[i] ;
```

```
Boucle : LF F1,0(R1)
LF F2,2048(R1)
FSUB F1,F2,F1
SF F1,0(R2)
ADDI R1,R1,4
ADDI R2,R2,4
ADDI R3,R3,-1
BNE R3,Boucle
```

On supposera que l'adresse de X[0] est initialement dans R1, que l'adresse de Y[0] est initialement dans R2. R3 contient initialement le nombre d'itérations de la boucle.

Question 1 : Quel est en nombre de cycles, le temps d'exécution par itération de la boucle originale sans et avec utilisation des instructions SIMD. ?

Sans SIMD : 6 cycles/itération
Avec SIMD : 7/4 = 1,75 cycles/itération

Sans déroulage, sans SIMD

	E0	E1	FA	FM
1 Boucle	LF F1, 0(R1)	LF F2, 2048(R1)		
2	ADDI R1,R1,4	ADDI R2,R2,4		
3	ADDI R3,R3,-1		FSUB F1,F1,F2	
4				
5				
6	SF -4(R2)	BNE R3, Boucle		

Sans déroulage, avec SIMD

	E0	E1	FA	FM
1 Boucle	PLF S1, 0(R1)	ADDI R3,R3,-4		
2	PLF S2, 2048(R1)	ADDI R1,R1,16		
3	ADDI R2,R2,16			
4			PFSUB S1,S1,S2	
5				
6				
7	PSF -16(R1)	BNE R3, Boucle		

Question 2

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 : 9/4 = 2,25 cycles/itération
Avec déroulage, avec SIMD : 13/ 16 = 0,8125 cycles/itération

Avec déroulage, sans SIMD

	E0	E1	FA	FM
1 Boucle	LF F1, 0(R1)	LF F2, 2048(R1)		
2	LF F3, 4(R1)	LF F4, 2052(R1)		
3	LF F5, 8(R1)	LF F6, 2056(R1)	FSUB F1,F1,F2	
4	LF F7, 12(R1)	LF F8, 2060(R1)	FSUB F3,F3,F4	
5	ADDI R1,R1,16	ADDI R2,R2,16	FSUB F5,F5,F6	
6	SF F1, -16(R2)	ADDI R3,R3,-4	FSUB F7,F7,F8	
7	SF F3, -12(R2)			
8	SF F5, -8(R2)			
9	SF F7, -4(R2)	BNE R3, Boucle		

Avec déroulage, SIMD

	E0	E1	FA	FM
1 Boucle	PLF S1, 0(R1)	ADDI R3,R3,-16		
2	PLF S2, 2048(R1)	ADDI R2,R2,64		
3	PLF S3, 4(R1)			
4	PLF S4, 2064(R1)		PFSUB S1,S1,S2	
5	PLF S5, 32(R1)			
6	PLF S6, 2080(R1)		PFSUB S3,S3,S4	
7	PLF S7, 48(R1)			
8	PLF S8, 2096(R1)		PFSUB S5,S5,S6	
9	PSF S1, -64(R2)	ADDI R1,R1,64		
10	PSF S3, -48(R2)		PFSUB S7,S7,S8	
11	PSF S5, -32(R2)			
12				
13	PSF S7, -16(R2)	BNE R3, Boucle		

Résumé :
Sans SIMD : 6 cycles/itération
Avec SIMD : 7/4 = 1,75 cycles/itération
Avec déroulage, sans SIMD : 9/4 = 2,25 cycles/itération
Avec déroulage, avec SIMD : 13/ 16 = 0,8125 cycles/itération

Question 3 : Pour la boucle ci-dessous, quelle est la condition pour pouvoir utiliser les instructions SIMD ?

```
float X[2*N], Y[N];
for (i=0 ; i<N ; i++)
      Y[i] = X[i+N] – X[i] ;
```

Il faut que N >=4 ; si N n'est pas un multiple de 4, le reste de la division N/4 doit être réalisé en scalaire

PREDICTEURS DE BRANCHEMENT

Soit les branchements 1 à 4 qui ont le comportement suivant sur 8 itérations (P pour pris et N pour non pris)

Itération	1	2	3	4	5	6	7	8
Branch 1	N	P	P	N	N	P	P	N
Branch 2	P	P	P	P	P	N	N	N
Branch 3	P	N	N	P	P	P	P	N
Branch 4	P	P	P	P	P	P	P	N

On dispose des prédicteurs suivants :

- statique toujours pris
- dynamique local 1 bit (initialisé à pris)
- dynamique local 2 bits (initialisé à fortement pris)

Statique
B1 : 4/8
B2 : 5/8
B3 : 5/8
B4 : 7/8

Prédicteur 1 bit

Itération	1	2	3	4	5	6	7	8
Branch 1	N	P	P	N	N	P	P	N
	P	N	P	P	N	N	P	P
Branch 2	P	P	P	P	P	N	N	N
	P	P	P	P	P	P	N	N
Branch 3	P	N	N	P	P	P	P	N
	P	P	N	N	P	P	P	P
Branch 4	P	P	P	P	P	P	P	N
	P	P	P	P	P	P	P	P

B1 : 3/8
B2 : 7/8
B3 : 5/8
B4 : 7/8

Prédicteur 2 bits

Itération	1	2	3	4	5	6	7	8
Branch 1	N	P	P	N	N	P	P	N
	FP	FP	FP	FP	fP	fN	fP	P
Branch 2	P	P	P	P	P	N	N	N
	FP	FP	FP	FP	FP	FP	fP	fN
Branch 3	P	N	N	P	P	P	P	N
	FP	FP	fP	fN	fP	FP	FP	FP
Branch 4	P	P	P	P	P	P	P	N
	FP	FP	FP	FP	FP	FP	FP	FP

B1 : 3/8
B2 : 6/8

Question 4) Quel est pour chaque branchement le meilleur prédicteur ? Quel est globalement le meilleur prédicteur pour les 4 branchements ?

B1 : statique
B2 : 1 bit
B3 : statique (moins coûteux)
B4 : statique (moins coûteux)
Globalement
1 bit (moins coûteux que 2 bits)

CACHES

Un processeur utilise un cache de données de 64 Ko, avec des lignes de 64 octets, à correspondance directe. Le cache utilise la réécriture avec écriture allouée (il y a des défauts de cache en écriture). Le processeur a des adresses sur 32 bits.
On considère l'extrait de programme C suivant, pour lequel les tableaux X et Y sont rangés successivement en mémoire à partir de l'adresse $1000\ 0000_H$ (adresse de X[0].)

```
float X[2*N], Y[N];
for (i=0 ; i<N ; i++)
      Y[i] = X[i+N] – X[i] ;
```

Question 5 : Quel est le nombre de défaut de caches par itération quand N = 512

6 bits adresse dans la ligne, 10 bits d'index (1K lignes) et 16 bits d'étiquette.
X[0] = 0x1000 0000 – ligne 0
X[512] = 0x1000 0800 =0001 0000 0000 0000 |0000 1000 00|00 0000 – ligne 32
Y[0] = 0x1000 1000 = 0001 0000 0000 0000 |0001 0000 00|00 0000 – ligne 64
Pas de défauts de conflits. Il y a donc 3 défauts toutes les 16 itérations.
Soit 3/16 défauts par itération

Question 6 : Pour quelle valeur de N (puissance de 2) a-t-on 3 défauts par itérations

Il y a 3 défauts par itération si l'index pour X[0], X[N] et X[2N] est le même,
Soit X[N] = 0x1001 0000 = X[0] + 2^{16} soit N=14. Toutes les valeurs 2^N, avec N≥ 14 provoquent 3 défauts par itération

PIPELINE LOGICIEL AVEC TMS 320C62

Le code assembleur TMS320C62 ci-dessous donne l'itération du pipeline logiciel pour un programme C.

```
LOOP:
            LDW   .D1   *A4++,A2          ;
||          ADD   .L1   A6,A7,A7          ;
||          ADD   .L2   B6,B7,B7          ;
||[A1]      B     .S2   LOOP              ;
||          MPY   .M1X  A2,A2,A6          ;
||          MPYH  .M2X  A2,A2,B6          ;
||[A1]      SUB   .S1   A1,1,A1     ;     decrement loop counter

            ADD   .L1X  A7,B7,A4          ;
```

Question 7 : Donner le code C correspondant au code assembleur.

```
short  X[N], Y[N], S,S1,S2 ;
```

```
S1=0; S2=0;
for (i = 0 ; i<N ; i+=2){
      S1 = X[i]*X[i];
      S2 = X[i+1]*X[i+1];}
S=S1+S2;
```

Question 8 : Quel est l'intervalle inter-itération (II) ? Justifier la valeur.

L'intervalle inter-itération est de 1. Les 7 instructions se répartissent sur 7 unités fonctionnelles (D1, L1,L2, S1, S2, M1, M2).

SIMD IA-32

Soit le programme P1

```
float X[512], Y[512], A[512];
for (i=0 ; i<512 ; i++)
      A[i] = (X[i] -Y[i])* 0.5 ;
```

Question 9 : En utilisant les intrinsics SIMD fournis en annexe, donner la version SIMD du programme P1. On rappelle que les float sont codés sur 32 bits et que les registres SSE ont une taille de 128 bits.

```
_m128 XS[128], YS[128], AS[128], S ;

S = SET4(0.5f) ;
For (i=0 ; i<128 ; i++)
      AS[i] = MULPS(SUBPS(XS[i],YS[i]), S);
```

Question 10 : Donner la suite des instructions pour faire la somme horizontale des 4 floats contenus dans une variable _m128 X.

```
X=HADDPS(X,X) ;
X=HADDPS (X,X) ;
```

Soit le programme P2

```
float Z[128], X[128], Y[128][128], S;
for (i=0 ; i<128 ; i++){
      S= 0.0f;
      for (j=0; j<128; j++)
            S+= X[j]*Y[i][j] ;
      Z[i]=S ;}
```

Question 11 : : En utilisant les intrinsics SIMD fournis en annexe, donner la version SIMD du programme P2.

```
_m128 Z[128],ZS[32], XS[32], YS[32][32], SS;
ZS=Z;
for (i=0;i<128;i++) {
      SS=SET4(0.0f);
      for (j=0;j<32;j++)
            SS=ADDPS (SS, MULPS (XS[j],YS[i][j]));
      SS=HADDPS(SS,SS); SS=HADDPS(SS,SS);
      MOVSS(Z[i], SS);
      }
```

ACCELERATION PARALLELE

Question 12 : A taille de programme constante, quelle est la fraction séquentielle maximale possible pour obtenir une efficacité parallèle de 60% sur 16 processeurs ?

$$\frac{1}{S + (1-S)/16} = \frac{0{,}6}{16}$$

Soit S = 0,4/9 = 4,44%

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

JEU D'INSTRUCTIONS (extrait)

LF	LF Fi, dép.(Ra)	2	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	1	E0 ou E1	Rd ←Ra + Rb
ADDI	ADDI Rd, Ra, IMM	1	E0 ou E1	Rd ←Ra + IMM-16 bits avec ES
SUB	SUB Rd,Ra, Rb	1	E0 ou E1	Rd ←Ra - Rb
FADD	FADD Fd, Fa, Fb	3	FA	Fd ←Fa + Fb
FSUB	FSUB Fd, Fa, Fb	3	FA	Fd ← Fa - Fb
FMAX	FMAX Fd, Fa, Fb	3	FA	Fd ← Max(Fa, Fb)
FMIN	FMIN Fd, Fa, Fb	3	FA	Fd ← Min(Fa, Fb)
FMUL	FMUL Fd, Fa, Fb	3	FM	Fd ←Fa x Fb
FDIV	FDIV Fd, Fa, Fb	20	FA	Fd ← Fa/Fb
BEQ	BEQ Ri, dépl	1	E1	si Ri=0 alors CP ← NCP + depl
BNE	BNE Ri, dépl	1	E1	si Ri≠0 alors CP ← NCP + depl

Table 1 : instructions disponibles

Le processeur a également 32 registres de 128 bits S0 à S7 pouvant contenir chacun 4 flottants simple précision et les instructions SIMD données dans la Table 2. 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
PFADD	PFADD Sd,Sa, Sb	3	FA	Sd ← Sa + Sb sur quatre « floats »
PFSUB	PFSUB Sd,Sa, Sb	3	FA	Sd ← Sa – Sb sur quatre « floats »
PFMUL	PFMUL Sd, Sa, Sb	4	FM	Sd ← Sa x Sb sur quatre « floats »
PFMULS	PMULS Sd, Sa, Fb	4	FM	SF ← Sa x 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
PFMAX	PMAX Sd,Sa, Sb	1	FA	Sd ← Sa max Sb : maximum sur 4 floats
PFMIN	PMIN Sd,Sa, Sb	1	FA	Sd ← Sa min Sb : minimum sur 4 floats

Table 2 : Instructions SIMD

ANNEXE 2 : Instructions SIMD IA-32 utilisables

ADDPS(a,b)	_mm_add_ps(a,b)	Quatre additions flottantes 32 bits
SUBPS(a,b)	_mm_sub_ps(a,b)	Quatre soustractions flottantes 32 bits
MULPS(a,b)	_mm_mul_ps(a,b)	Quatre multiplications flottantes 32 bits
DIVPS(a,b)	_mm_div_ps (a,b)	Quatre divisions flottantes 32 bits
SET4	_mm_set_ps1 (float w)	Quatre fois le flottant 32 bits w dans un mot de 128 bits
HADDPS(a,b)	_mm_hadd_ps(a,b)	b3+b2|b1+b0|a3+a2|a1+a0
MOVSS(a,b)	_mm_store_ss(*a,b)	Mem32(a) ← 32 bits poids faible de b (sur 128 bits)