

Master professionnel Informatique

Optimisations pour graphique et multimédia

Examen Mars 2006
1H ; tous documents autorisés

Question 1

Pour les boucles suivantes, indiquer celles qui peuvent utiliser les instructions SIMD (boucles « vectorisables ») et celles qui ne le peuvent pas en précisant brièvement pourquoi.

```
int i, j ;  
float X[N], Y[N] ;
```

Boucle 1

```
for (i=0 ; i<N ; i++)  
    for (j=0 ; j<N ; j++)  
        Y[i][j] = X[i][j] + X[i][j-1];
```

Boucle 2

```
for (i=0 ; i<N ; i++)  
    for (j=0 ; j<N ; j++)  
        Y[i][j] = Y[i][j] + Y[i][j-1];
```

Boucle 3

```
for (i=0 ; i<N ; i++)  
    for (j=0 ; j<N ; j++)  
        Y[j][i] = Y[j][i] + X[j-1][i];
```

Question 2

Soit le code suivant pour un filtre 3x3 (EEMBC High-Pass Gray Filter)

```
int i, j;  
unsigned char X[size][size], X[size][size];  
short temp;  
for(i=1; i<size-1; i++) {  
    for(j=1; j<size-1; j++) {  
        tmp = (short) (F11* X[i-1][j-1]+F21* X[i-1][j]+F31* X[i-1][j+1]  
                    + F12* X[i][j-1]+F22* X[i][j]+F32* X[i][j+1]  
                    + F13* X[i+1][j-1]+F23* X[i+1][j]+F33* X[i+1][j+1]);  
        Y[i][j]= (unsigned char) (tmp/256);  
    }  
}
```

Figure 1: Filtre 3 x 3

F22 a la valeur 255 et tous les autres coefficients ont la valeur -28

Ecrire une version C scalaire optimisée du code de la figure 1.

Question 3

Soit le filtre passe haut suivant que l'on applique sur une image 128 * 128 en niveaux de gris (les pixels sont des octets non signés) :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 2 : Filtre 3 x 3

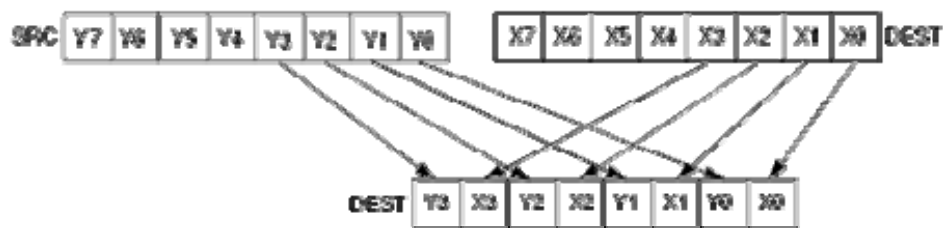
En utilisant les intrinsèques des instructions SIMD IA-32, écrire une version SIMD optimisée du filtre de la figure 2.

On pourra utiliser les « define » vus en cours et en TP.

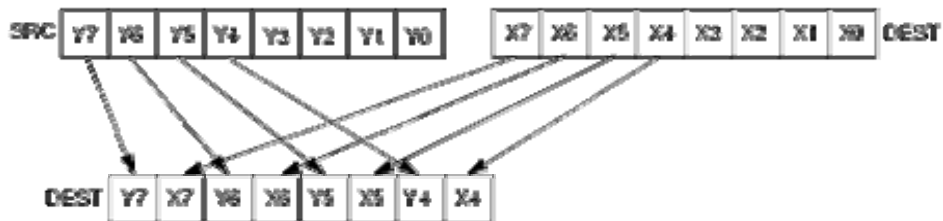
On appellera X[128][128] l'image source et Y[128][128] l'image destination.

ANNEXE : Instructions SIMD IA-32 utilisables

MOVDQA	_mm_load_si128 (*p)	Chargement aligné de 128 bits
MOVDQA	_mm_store_si128 (*p,a)	Rangement aligné de 128 bits
PACKUSWB	_mm_packus_epi16 (m1,m2)	8 shorts de m1 dans les 8 octets bas et 8 shorts de m2 dans les 8 octets haut (compactage avec saturation)
PADDUB	_mm_add_epu8 (a,b)	Addition non signée saturée sur octets (16 x 8 bits)
PADDUW	_mm_add_epu16 (a,b)	Addition non signée saturée sur shorts (8 x 16 bits)
POR	_mm_or_si128 (a, b)	Ou logique parallèle
PSRLDQ	_mm_srli_si128 (a, imm)	Décalage logique gauche de « imm » octets
PSRLW	_mm_srli_epi16 (m1, N)	Décalage à gauche de N positions de chaque short (8 x 16) en entrant des 0 à droite.
PSSLDQ	_mm_slli_si128 (a, imm)	Décalage logique droite de « imm » octets
PSUBB	_mm_sub_epi8 (a,b)	Soustraction signée sur octets (16 x 8 bits)
PSUBUSB	_mm_sub_epu8 (a,b)	Soustraction non signée saturée sur octets (16 x 8 bits)
PSUBUSW	_mm_sub_epu16 (a,b)	Soustraction non signée saturée sur shorts (8 x 16 bits)
PSUBW	_mm_sub_epi16 (a,b)	Soustraction signée sur shorts (8 x 16 bits)
PUNPCKHBW	_mm_unpacklo_epi8 (m1, m2)	Entrelace les octets (haut) de la destination et la source
PUNPCKHWD	_mm_unpacklo_epi16(m1, m2)	Entrelace les shorts (haut) de la destination et la source
PUNPCKLBW	_mm_unpacklo_epi8 (m1, m2)	Entrelace les octets (bas) de la destination et la source
PUNPCKLWD	_mm_unpacklo_epi16 (m1, m2)	Entrelace les shorts (bas) de la destination et la source
PXOR	_mm_xor_si128 (a, b)	Ou exclusif parallèle



PUNPCKLBW



PUNPCKHBW