

# QuestionnaireGroupe2

Merci de prendre part à cette étude qui vise à comprendre l'influence que peut avoir la visualisation sur la compréhension du comportement de logiciel. Vous avez à répondre à 12 questions dont les réponses vont de remplissages automatiques à quelques calculs que vous pouvez faire mentalement. Vous pouvez aussi vous servir d'une calculatrice ou d'une feuille de calcul Excel. Si vous le souhaitez, vous pouvez répondre de façon anonyme à ce questionnaire en fournissant un pseudonyme à la place de votre nom. Cette étude a obtenu une certification éthique du Comité d'éthique de la recherche avec les êtres humains de l'université TELUQ (CER-TELUQ) numéro 2022-08 du 12 avril 2022.

## A- Renseignements personnels, date et heure du début de remplissage du questionnaire

Q1- Nom ou pseudonyme

MND

Q2-Date :

2022-05-23

Q3-Heure du début :

16:02

## C- Compréhension du comportement des instructions vectorielles `_mm512_mask_add_ps` et `_mm_shuffle_epi32`

### 1- Instruction vectorielle `_mm512_mask_add_ps`

Avant de répondre aux questions Q7 et Q8, observez attentivement la figure ci-dessous qui est une capture d'écran du prototype SIMD Giraffe, disponible en ligne sur <https://github.com/pmmtang/SIMDGiraffe>. Cette capture d'écran est divisée en trois cadrans.

Sur le cadran de gauche il y a une description de l'instruction fournie par Intel®.

Sur le cadran d'en bas, il y a la traduction graphique de cette description. Cette traduction consiste à afficher pour chacun de ces vecteurs, ses champs (ou coordonnées). Nous utilisons les lettres indexées de l'alphabet ( $A_0, A_1, \dots, B_0, B_1, \dots$ ) inscrites à l'intérieur de rectangles bleus pour désigner ces champs. Par exemple les champs (coordonnées) du vecteur src sont  $A_0, A_1, \dots$  ce qui veut dire que  $src_0 = A_0, src_1 = A_1, \dots$ ; ainsi de suite pour les autres vecteurs opérandes; quant au vecteur résultat, les champs (coordonnées) de r sont  $E_0, E_1, \dots$  ce qui veut dire que  $r_0 = E_0, r_1 = E_1, \dots$ . La description graphique est précédée sur la ligne, à gauche du signe de l'égalité (=), par le nom du vecteur en question (src, k, a, b, r) et son type (`_mm512`, `_mmask16`, `_mm512`, `_mm512`, `_mm512`).

Sur le cadran de droite, il y a la description visuelle des liens entre chaque champ (ou coordonnée) du vecteur résultat r et les champs (ou coordonnées) des vecteurs opérandes utilisés pour calculer ce champ (ou cette coordonnée). Cette description consiste comme on peut le remarquer, à donner pour chaque champ du vecteur résultat r la formule qui permet de calculer ce champ à partir des champs opérandes utilisés pour effectuer ce calcul. Par exemple on peut voir sur ce cadran que  $r_0 = E_0 = (1-B_0) \times A_0 + B_0 \times (C_0 + D_0)$ ,  $r_1 = E_1 = (1-B_1) \times A_1 + B_1 \times (C_1 + D_1)$ , ...

Vous ne devez utiliser que les explications fournies (vous pouvez naturellement consulter le site d'Intel® ou le site de [SIMDGiraffe](https://github.com/pmmtang/SIMDGiraffe)), mais ne faites pas recours à d'autres ressources (par exemple recherche sur Google, autres documents, etc.)

#### Choose SIMD Instruction

`_mm512_mask_add_ps`

`_mm512_mask_add_ps(_mm512 src, __mmask16 k, _mm512 a, _mm512 b)`

#### Synopsis

`_mm512_mask_add_ps(_mm512 src, __mmask16 k, _mm512 a, _mm512 b)`

b)

#include <immintrin.h>

Instruction: vaddps zmm {k}, zmm, zmm

CPUID Flags: AVX512F/KNCNI

#### Description

Add packed single-precision (32-bit) floating-point elements in "a" and "b", and store the results in "dst" using writemask "k" (elements are copied from "src" when the corresponding mask bit is not set).

#### Operation

FOR j := 0 to 15

i := j\*32

IF k[j]

dst[i+31:i] := a[i+31:i] + b[i+31:i]

ELSE

dst[i+31:i] := src[i+31:i]

FI

ENDFOR

dst[MAX:512] := 0

#### Novice view

How to compute these fields:

$$\begin{aligned} E_{15} &= (1-B_{15}) \times A_{15} + B_{15} \times (C_{15} + D_{15}) & E_{14} &= (1-B_{14}) \times A_{14} + B_{14} \times (C_{14} + D_{14}) \\ E_{13} &= (1-B_{13}) \times A_{13} + B_{13} \times (C_{13} + D_{13}) & E_{12} &= (1-B_{12}) \times A_{12} + B_{12} \times (C_{12} + D_{12}) \\ E_{11} &= (1-B_{11}) \times A_{11} + B_{11} \times (C_{11} + D_{11}) & E_{10} &= (1-B_{10}) \times A_{10} + B_{10} \times (C_{10} + D_{10}) \\ E_9 &= (1-B_9) \times A_9 + B_9 \times (C_9 + D_9) & E_8 &= (1-B_8) \times A_8 + B_8 \times (C_8 + D_8) \\ E_7 &= (1-B_7) \times A_7 + B_7 \times (C_7 + D_7) & E_6 &= (1-B_6) \times A_6 + B_6 \times (C_6 + D_6) \\ E_5 &= (1-B_5) \times A_5 + B_5 \times (C_5 + D_5) & E_4 &= (1-B_4) \times A_4 + B_4 \times (C_4 + D_4) \\ E_3 &= (1-B_3) \times A_3 + B_3 \times (C_3 + D_3) & E_2 &= (1-B_2) \times A_2 + B_2 \times (C_2 + D_2) \\ E_1 &= (1-B_1) \times A_1 + B_1 \times (C_1 + D_1) & E_0 &= (1-B_0) \times A_0 + B_0 \times (C_0 + D_0) \end{aligned}$$

[return to expert view](#)

operator =	+	x	-	/	mov	:int	exp	ln	(	)	ldx	inv				
__m512 src =	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
__mmask16 k =	B <sub>15</sub>	B <sub>14</sub>	B <sub>13</sub>	B <sub>12</sub>	B <sub>11</sub>	B <sub>10</sub>	B <sub>9</sub>	B <sub>8</sub>	B <sub>7</sub>	B <sub>6</sub>	B <sub>5</sub>	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>
__m512 a =	C <sub>15</sub>	C <sub>14</sub>	C <sub>13</sub>	C <sub>12</sub>	C <sub>11</sub>	C <sub>10</sub>	C <sub>9</sub>	C <sub>8</sub>	C <sub>7</sub>	C <sub>6</sub>	C <sub>5</sub>	C <sub>4</sub>	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>
__m512 b =	D <sub>15</sub>	D <sub>14</sub>	D <sub>13</sub>	D <sub>12</sub>	D <sub>11</sub>	D <sub>10</sub>	D <sub>9</sub>	D <sub>8</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
__m512 r =	E <sub>15</sub>	E <sub>14</sub>	E <sub>13</sub>	E <sub>12</sub>	E <sub>11</sub>	E <sub>10</sub>	E <sub>9</sub>	E <sub>8</sub>	E <sub>7</sub>	E <sub>6</sub>	E <sub>5</sub>	E <sub>4</sub>	E <sub>3</sub>	E <sub>2</sub>	E <sub>1</sub>	E <sub>0</sub>

**Q7- Après avoir bien observé la figure ci-dessus dite ce que fait l'instruction**

**`_mm512_mask_add_ps` en effectuant le calcul suivant: étant donnés  $\text{src}=(1, 3, 4, 1, 2, 5, 4, 1, 2, 3, 4, 1, 1, 3, 4, 1)$  ;  $\text{k}=(1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0)$  ;  $\text{a}=(6, 1, 2, 3, 1, 4, 5, 1, 2, 3, 4, 1, 3, 1, 2, 1)$  ;  $\text{b}=(6, 1, 2, 3, 1, 4, 5, 1, 2, 3, 4, 1, 3, 1, 2, 1)$  . Calculer  $\text{r} = \text{\_mm512\_mask\_add\_ps}(\text{src}, \text{k}, \text{a}, \text{b})$**   
 $\text{r} = (12, 3, 4, 6, 2, 5, 4, 2, 2, 6, 8, 1, 1, 3, 4, 1)$

**Q8- Toujours à l'aide de la figure ci-dessus donnez une formule générale de calcul des coordonnées de  $\text{r}(\text{ri})$  en fonction de celles de  $\text{src}(\text{srci})$ ,  $\text{k}(\text{ki})$ ,  $\text{a}(\text{ai})$  et  $\text{b}(\text{bi})$ .  $\text{ri} = ?$**

$\text{ri} = \text{Ei} = (1 - \text{Bi}) \times \text{Ai} + \text{Bi} \times (\text{Ci} + \text{Di}) = (1 - \text{ki}) \times \text{srci} + \text{ki} \times (\text{ai} + \text{bi})$

## 2- Instruction vectorielle `_mm_shuffle_epi32`

Avant de répondre aux questions Q9 et Q10, observez attentivement la figure ci-dessous qui est une capture d'écran du prototype SIMD Giraffe, disponible en ligne sur

<https://github.com/pmntang/SIMDGiraffe>. Cette capture d'écran est divisée en trois cadrans.

Sur le cadran de gauche il y a une description de l'instruction fournie par Intel®.

Sur le cadran d'en bas, il y a la traduction graphique de cette description. Cette traduction consiste à afficher pour chacun de ces vecteurs, ses champs (ou coordonnées). Nous utilisons les lettres indexées de l'alphabet (A0, A1, ... B0, B1, ..., ...) inscrites à l'intérieur de rectangles bleus pour désigner ces champs. Ainsi les champs (coordonnées) du vecteur a sont A0, A1, A2, A3, ce qui veut dire que  $\text{a}_0 = \text{A}_0$ ,  $\text{a}_1 = \text{A}_1$ ,  $\text{a}_2 = \text{A}_2$ ,  $\text{a}_3 = \text{A}_3$  ; les champs (coordonnées) du vecteur imm8 sont B0, B1, B2, B3, ce qui veut dire que

imm80 = B0, imm81 = B1, imm82 = B2, imm83 = B3 ; quant au vecteur résultat, les champs (coordonnées) de r sont C0, C1, C2, C3; ce qui veut dire que r0 = C0, r1 = C1, r2 = C2, r3 = C3. La description graphique est précédée sur la ligne, à gauche du signe de l'égalité (=), par le nom du vecteur en question (a, imm8, r) et son type (\_\_m128i, int, \_\_m128i).

Sur le cadran de droite, il y a la description visuelle des liens entre chaque champ (ou coordonnée) du vecteur résultat r et les champs (ou coordonnées) des vecteurs opérands utilisés pour calculer ce champ (ou cette coordonnée). Cette description consiste comme on peut le remarquer, à donner pour chaque champ du vecteur résultat r la formule qui permet de calculer ce champ à partir des champs opérands utilisés pour effectuer ce calcul. On peut ainsi voir sur ce cadran que r0= C0 = AB0; r1 = C1 = AB1; r2 = C2 = AB2; r3 = C3 = AB3.

Vous ne devez utiliser que les explications fournies (vous pouvez naturellement consulter le site d'Intel® ou le site de [SIMDGiraffe](#) ), mais ne faites pas recours à d'autres ressources (par exemple recherche sur Google, autres documents, etc.).

Choose SIMD Instruction

\_mm\_shuffle\_epi32

\_m128i \_mm\_shuffle\_epi32 (\_m128i a, int imm8)

Synopsis

```

_m128i _mm_shuffle_epi32 (_m128i a, int imm8)
#include <emmintrin.h>
Instruction: pshufd xmm, xmm, imm
CPUID Flags: SSE2

```

Description

Shuffle 32-bit integers in "a" using the control in "imm8", and store the results in "dst".

Operation

```

DEFINE SELECT4(src, control) {
    CASE(control[1:0]) OF
    0:    tmp[31:0] := src[31:0]
    1:    tmp[31:0] := src[63:32]
    2:    tmp[31:0] := src[95:64]
    3:    tmp[31:0] := src[127:96]
    ESAC
    RETURN tmp[31:0]
}
dst[31:0] := SELECT4(a[127:0], imm8[1:0])
dst[63:32] := SELECT4(a[127:0], imm8[3:2])
dst[95:64] := SELECT4(a[127:0], imm8[5:4])
dst[127:96] := SELECT4(a[127:0], imm8[7:6])

```

**Novice view**

How to compute these fields:

$C_3 = A_{B_3}$   $C_2 = A_{B_2}$   $C_1 = A_{B_1}$   $C_0 = A_{B_0}$

return to expert view

---

operator = + x - / mov :(int) exp ln ( ) ldx inv

\_m128i a = A<sub>3</sub> A<sub>2</sub> A<sub>1</sub> A<sub>0</sub>

int imm8 = B<sub>3</sub> B<sub>2</sub> B<sub>1</sub> B<sub>0</sub>

\_m128i r = C<sub>3</sub> C<sub>2</sub> C<sub>1</sub> C<sub>0</sub>

**Q9- Après avoir bien observé la figure ci-dessus dite ce que fait l'instruction `_mm_shuffle_epi32` en**

effectuant le calcul suivant: étant donnés  $a=(6, 7, 4, 3)$  ;  $imm8=(0, 1, 2, 3)$  . Calculez  $r = \_mm\_shuffle\_epi32(a, imm8)$   
 $r = (3, 4, 7, 6)$

**Q10-** Toujours à l'aide de la figure ci-dessus, donnez une formule générale de calcul des coordonnées de  $r(r_i)$  en fonction de celles de  $a(a_i)$  et  $imm8(imm8_i)$ .  $r_i=?$   
 $r_i=C_i=A_j=a_j$ , avec  $j=B_i=imm8_i$

## B- Connaissances préliminaires

### I- Connaissance de l'algèbre et de l'espace vectoriel

Considérons l'espace vectoriel réel  $R^3$ . Pour  $A, B, C, Res1, Res2$ , cinq vecteurs de  $R^3$  tels que  $A=(a_1, a_2, a_3)$ ,  $B=(b_1, b_2, b_3)$ ,  $C=(c_1, c_2, c_3)$ ,  $Res1=(x_1, x_2, x_3)$ ,  $Res2=(y_1, y_2, y_3)$  on définit  $vectSum(A,B,C)=Res1$  et  $vectProd(A,B,C)=Res2$  par

$$\begin{cases} x_1 = a_1 - b_1 + c_1 \\ x_2 = a_2 - b_2 + c_2 \\ x_3 = a_3 - b_3 + c_3 \end{cases} \text{ and } \begin{cases} y_1 = b_1 \times (a_1 - c_1) + c_1 \\ y_2 = b_2 \times (a_2 - c_2) + c_2 \\ y_3 = b_3 \times (a_3 - c_3) + c_3 \end{cases}$$

On suppose maintenant que  $A=(1,0,1)$ ;  $B=(1,1,0)$  ;  $C=(0,1,1)$ .

**Q4- Calculez chacun des vecteurs  $Res1$  et  $Res2$ :  $Res1=?$   $Res2=?$**   
 $Res1=(0,0,2)$  ;  $Res2=(1,0,1)$ .

**Q5- Donnez une formule générale de calcul des coordonnées de  $Res1(x_i)$  et de  $Res2(y_i)$  en fonction de celles de  $A(a_i)$ ,  $B(b_i)$  et  $C(c_i)$ .  $x_i=?$  ;  $y_i=?$**

## II- Connaissance du langage C

Considérons la fonction  $f$  suivante en C: `int f (int x, int y) {return x-y;}`.

**Q6- Déterminez en C deux instructions (soit `instruction1` et `instruction2`) qui permettent de déclarer trois variables entiers  $a, b, c$  et de placer dans  $c$  la différence de  $a$  et  $b$  à l'aide de la fonction  $f$ .**  
**Instruction1: ? Instruction2: ?**  
Instruction1: `int c, a, b;` Instruction2: `{return c=f(a-b);}`

## D- Heure de la fin de remplissage du questionnaire et commentaires

**Q11- Heure de fin:**  
20:10

**Q12- Autres commentaires et remarques:**  
Complicé mais ça va