

# QuestionnaireGroup2

Thank you for taking part in this study which aims to understand the influence that visualization can have on the understanding of software behavior. You are kindly requested to answer 12 questions whose answers range from automatic completions to some calculations that you can do mentally. You can also use a calculator or an Excel spreadsheet. If you wish, you can answer this questionnaire anonymously by providing a pseudonym instead of your name. This study has received ethical certification from the Ethics Committee for Research with Human Beings of TELUQ University (CER-TELUQ) number 2022-08 of April 12, 2022.

## A- Personal information, start date and start time of filling out the questionnaire

### Q1- Name or pseudonym

MarkChristensen

### Q2-Date :

2022-05-30

### Q3-Start time:

3:09 PM

## C- Understanding the behavior of the vector instructions \_\_mm512\_mask\_add\_ps and \_\_mm\_shuffle\_epi32

### 1- Vector instruction \_\_mm512\_mask\_add\_ps

Before answering questions Q7 and Q8, carefully observe and try to understand the figure below which is a screenshot of the SIMD Giraffe prototype, available online at <https://github.com/pmntang/SIMDGiraffe>. This screenshot is divided into three dials.

On the left dial there is a description of the instruction provided by Intel®.

On the lower dial, there is the graphical translation of this description. This translation consists in displaying for each of these vectors, its fields (or coordinates). We use the letters of the alphabet with subscripts ( $A_0$ ,  $A_1$ , ...,  $B_0$ ,  $B_1$ , ..., ...) written inside blue rectangles to designate these fields. The graphic description is preceded on the line, to the left of the equality sign ( $=$ ), by the name of the vector in question ( $src$ ,  $k$ ,  $a$ ,  $b$ ,  $r$ ) and its type (`__m512`, `__mmask16`, `__m512`, `__m512`, `__m512`).

On the right-hand dial there is a visual description of the links between each field (or coordinates) of the result vector  $r$  and the fields (or coordinates) of the operand vectors used to calculate this field (or this coordinate). This description consists, as we can see, in giving for each field of the vector result  $r$  the calculation formula of that field from the operand fields used to carry out this calculation.

### 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- After observing the figure above, say what the `_mm512_mask_add_ps` instruction does by performing the following calculation: given `src=(1, 3, 4, 1, 2, 5, 4, 1, 2, 3, 4, 1, 1, 3, 4, 1)`; `k=(1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0)`; `a=(6, 1, 2, 3, 1, 4, 5, 1, 2, 3, 4, 1, 3, 1, 2, 1)`; `b=(6, 1, 2, 3, 1, 4, 5, 1, 2, 3, 4, 1, 3, 1, 2, 1)`. Calculate `r = _mm512_mask_add_ps(src, k, a, b)`  
`r = (12, 3, 4, 6, 2, 5, 4, 2, 2, 6, 8, 1, 1, 3, 4, 1)`**

**Q8- Using the figure above again, give a general formula for calculating the coordinates of `r(ri)` as function of those of `src(srci)`, `k(ki)`, `a(ai)` and `b(bi)` ). `ri=?`  
`ri=Ei=(1-Bi) x Ai+Bi x (Ci+Di)=(1-ki) x srci+ki x (ai+bi)`**

## 2-Vector instruction `_mm_shuffle_epi32`

Before answering questions *Q9* and *Q10*, carefully observe and try to understand the figure below which is a screenshot of the SIMD Giraffe prototype, available online at <https://github.com/pmntang/SIMDGiraffe>. This screenshot is divided into three dials.

On the left dial there is a description of the instruction provided by Intel®.

On the lower dial, there is the graphical translation of this description. This translation consists in displaying for each of these vectors, its fields (or coordinates). We use the letters of the alphabet with subscripts (A<sub>0</sub>, A<sub>1</sub>, ... B<sub>0</sub>, B<sub>1</sub>, ..., ...) written inside blue rectangles to designate these fields. The description graphic is preceded on the line, to the left of the equality sign (=), by the name of the vector in question (a, imm8, r)

and its type (\_\_m128i, int, \_\_m128i).

On the right-hand dial there is a visual description of the links between each field (or coordinate) of the result vector r and the fields (or coordinates) of the operand vectors used to calculate this field (or this coordinate). This description consists, as we can see, in giving for each field of the vector result r the calculation formula of that field from the operand fields used to carry out this calculation.

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-** After observing the figure above, say what the `_mm_shuffle_epi32` instruction does by performing the following calculation: given `a=(6, 7, 4, 3)`; `imm8=(0, 1, 2, 3)` . Calculate `r = _mm_shuffle_epi32(a, imm8)`  
`r = (3, 4, 7, 6)`

**Q10-** Using the figure above again, give a general formula for calculating the coordinates of `r(ri)` as function of those of `a(ai)` and `imm8(imm8i)`. `ri=?`  
`ri=Ci=Aj=aj`, where `j=Bi=imm8i`

## B- Preliminary knowledge

### I- Knowledge of algebra and vector space

Consider the real vector space  $R^3$ . For  $A, B, C, \text{Res1}, \text{Res2}$ , five vectors of  $R^3$  such that  $A=(a_1, a_2, a_3)$ ,  $B=(b_1, b_2, b_3)$ ,  $C=(c_1, c_2, c_3)$ ,  $\text{Res1}=(x_1, x_2, x_3)$ ,  $\text{Res2}=(y_1, y_2, y_3)$  we define  $\text{vectSum}(A,B,C)=\text{Res1}$  and  $\text{vectProd}(A,B,C)=\text{Res2}$  by

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

Now let's assume that  $A=(1, 0, 1)$  ;  $B=(1, 1, 0)$  ;  $C=(0, 1, 1)$ .

**Q4- Calculate each of the Res1 and Res2 vectors: Res1= ? Res2=?**

$\text{Res1}=(0, 0, 2)$  ;  $\text{Res2}=(1, 0, 1)$ .

**Q5- Give a general formula for calculating the coordinates of Res1(xi) and Res2(yi) as a function of those of A (ai), B (bi) and C (ci). xi=? yi=?**

$x_i = a_i - b_i + c_i$ ;  $y_i = b_i \times (a_i - c_i) + c_i$ .

### II- Knowledge of the C language

Consider the following function f in C: `int f (int x, int y) {return x-y;}`.

**Q6- Choose the two instructions in C (that is, instruction1 and instruction2) which allow you to declare three integer variables a, b, c and to place in c the difference between a and b using the function f. instruction1: ? instruction2: ?**

Instruction1: `int c, a, b;` Instruction2: `c=f(a,b);`

## D- End time of the questionnaire completion and comments

**Q11- End time:**

3:45 PM

**Q12- Other comments and remarks:**