Laboratorium programowania niskopoziomowego

LAB 15 – Operacje przy użyciu instrukcji wektorowych AVX wykorzystanie instrukcji warunkowych.

O P I S

Forma zajęć: przykładowe zadania do analizy

Student na zajęciach powinien:

A. Wiedzieć jak działa kod i go rozumieć
B. Odpowiedzieć na pytania prowadzącego zajęcia

Zadanie 1: Posiadając trzy dwuwymiarowe dynamiczne tablice należy wyzerować tablicę pierwszą tylko dla tych elementów, które będą mniejsze lub równe od elementów im odpowiadającym z tablicy drugiej. Natomiast w tablicy drugiej należy wyzerować elementy mniejsze od elementów w tablicy pierwszej. Następnie do tablicy trzeciej wstępnie wyzerowanej dodać poszczególne wartości z tablicy pierwszej i drugiej, a na koniec finalnie tablicę trzecią wymnożyć przez stałą X. Wszystkie elementy typu double.

```
;Ify(double **tab1, double **tab2, double **tab3, int n, int m, double x);
Ify1 proc uses rsi rdi, tab1:ptr, tab2:ptr, tab3:ptr, n:dword, m:dword, x:qword
    local m1:qword
    mov m1, -1; FFFFFFFFFFFFFFFF
    movsxd r11, m;
    vbroadcastsd    ymm0, x;
    vbroadcastsd    ymm5, m1;
    petlaN:
        mov rsi, [rcx+8*r9-8]; tab1
        mov rdi, [rdx+8*r9-8]; tab2
        mov rax, [r8+8*r9-8];  tab3
        mov r10, r11; m
        sal r10, 3; m*8
        petlaM:
        sub r10, 32; -32
        vmovupd ymm1, ymmword ptr[rsi + r10]; tab1[][]
        vmovupd ymm2, ymmword ptr[rdi + r10]; tab2[][]
        vcmpgtpd ymm3, ymm1, ymm2 ; => vcmppd ymm3, ymm1, ymm2, 14;
        vpxor ymm4, ymm3, ymm5;
; XOR z (FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF)
        vandpd ymm1, ymm1, ymm3; "zerowanie" tab1
        vandpd ymm2, ymm2, ymm4; "zerowanie" tab2

        vaddpd ymm3, ymm1, ymm2; tab3[][] = tab1[][] + tab2[][]
        vmulpd ymm3, ymm3, ymm0; tab3[][] * X

        vmovupd ymmword ptr[rsi + r10], ymm1; tab1[][]
        vmovupd ymmword ptr[rdi + r10], ymm2; tab2[][]
        vmovupd ymmword ptr[rax + r10], ymm3; tab3[][]

        jns petlaM
        dec r9;
    jnz petlaN
    ret
Ify1 endp
```

PRZYPOMINAJKA

```
//    /* Compare */
//    #define _CMP_EQ_OQ    0x00  0  /* Equal (ordered, non-signaling)  */
//    #define _CMP_LT_OS    0x01  1  /* Less-than (ordered, signaling)  */
//    #define _CMP_LE_OS    0x02  2  /* Less-than-or-equal (ordered, signaling)  */
//    #define _CMP_UNORD_Q  0x03  3  /* Unordered (non-signaling)  */
//    #define _CMP_NEQ_UQ   0x04  4  /* Not-equal (unordered, non-signaling)  */
//    #define _CMP_NLT_US   0x05  5  /* Not-less-than (unordered, signaling)  */
//    #define _CMP_NLE_US   0x06  6  /* Not-less-than-or-equal (unordered, signaling)  */
//    #define _CMP_ORD_Q    0x07  7  /* Ordered (non-signaling)   */
//    #define _CMP_EQ_UQ    0x08  8  /* Equal (unordered, non-signaling)  */
//    #define _CMP_NGE_US   0x09  9  /* Not-greater-than-or-equal (unordered, signaling)  */
//    #define _CMP_NGT_US   0x0a  10 /* Not-greater-than (unordered, signaling)  */
//    #define _CMP_FALSE_OQ 0x0b  11 /* False (ordered, non-signaling)  */
//    #define _CMP_NEQ_OQ   0x0c  12 /* Not-equal (ordered, non-signaling)  */
//    #define _CMP_GE_OS    0x0d  13 /* Greater-than-or-equal (ordered, signaling)  */
//    #define _CMP_GT_OS    0x0e  14 /* Greater-than (ordered, signaling)  */
//    #define _CMP_TRUE_UQ  0x0f  15 /* True (unordered, non-signaling)  */
//    #define _CMP_EQ_OS    0x10  16 /* Equal (ordered, signaling)  */
//    #define _CMP_LT_OQ    0x11  17 /* Less-than (ordered, non-signaling)  */
//    #define _CMP_LE_OQ    0x12  18 /* Less-than-or-equal (ordered, non-signaling)  */
//    #define _CMP_UNORD_S  0x13  19 /* Unordered (signaling)  */
//    #define _CMP_NEQ_US   0x14  20 /* Not-equal (unordered, signaling)  */
//    #define _CMP_NLT_UQ   0x15  21 /* Not-less-than (unordered, non-signaling)  */
//    #define _CMP_NLE_UQ   0x16  22 /* Not-less-than-or-equal (unordered, non-signaling)  */
//    #define _CMP_ORD_S    0x17  23 /* Ordered (signaling)  */
//    #define _CMP_EQ_US    0x18  24 /* Equal (unordered, signaling)  */
//    #define _CMP_NGE_UQ   0x19  25 /* Not-greater-than-or-equal (unordered, non-signaling)*/
//    #define _CMP_NGT_UQ   0x1a  26 /* Not-greater-than (unordered, non-signaling)  */
//    #define _CMP_FALSE_OS 0x1b  27 /* False (ordered, signaling)  */
//    #define _CMP_NEQ_OS   0x1c  28 /* Not-equal (ordered, signaling)  */
//    #define _CMP_GE_OQ    0x1d  29 /* Greater-than-or-equal (ordered, non-signaling)  */
//    #define _CMP_GT_OQ    0x1e  30 /* Greater-than (ordered, non-signaling)  */
//    #define _CMP_TRUE_US  0x1f  31 /* True (unordered, signaling)  */
```

Porównanie gdy jeden z operandów jest NaN, ***ordered*** vs ***unordered***:

Porównanie w trybie ***Ordered*** dla liczb typu NaN da: **false**

- _CMP_EQ_UQ of 1.0 and 1.0 gives true (vanilla equality).
- _CMP_EQ_UQ of NaN and 1.0 gives false.
- _CMP_EQ_UQ of 1.0 and NaN gives false.
- _CMP_EQ_UQ of NaN and NaN gives false.

Porównanie w trybie ***Unordered*** dla liczb typu NaN da: **true**

- _CMP_EQ_UQ of 1.0 and 1.0 gives true (vanilla equality).
- _CMP_EQ_UQ of NaN and 1.0 gives true.
- _CMP_EQ_UQ of 1.0 and NaN gives true.
- _CMP_EQ_UQ of NaN and NaN gives true.

Różnica między **signalling** a **non-signalling** wpływa tylko na wartość **MXCSR**. Aby zaobserwować efekt, należałoby wyczyścić MXCSR, wykonać jedno lub więcej porównań, a następnie odczytać wartość z MXCSR.

Zadanie 2 Posiadając tablicę dwuwymiarową dynamiczną wypełniona losowymi liczbami typu double podnieś każdą wartość tej tablicy do kwadratu jeżeli początkowa wartość danego elementu jest większa niż X

```
;Ify(double **tab, int n, int m, double X); // RCX = **; RDX = N; R8 = M; XMM3 = X
Ify2 proc
    local m1:qword
    mov m1, -1; FFFFFFFFFFFFFFFF
    vbroadcastsd    ymm0, xmm3;
    vbroadcastsd    ymm4, m1;
    petlaN:
        mov rax, [rcx+8*rdx-8]; tab1
        mov r9, r8; m
        sal r9, 3; m*8
        petlaM:
        sub r9, 32; -32
        vmovupd ymm1, ymmword ptr[rax + r9]; tab[][]

        vcmppd ymm2, ymm1, ymm0, 1; 1-L;
        vandpd ymm3, ymm1, ymm2; "zerowanie"
        vmulpd ymm3, ymm3, ymm3;  ^2

        vpxor ymm2, ymm2, ymm4
; XOR z (FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF)
        vandpd ymm1, ymm1, ymm2; "zerowanie"

        vaddpd ymm1, ymm1, ymm3; złączenie

        vmovupd ymmword ptr[rax + r9], ymm1; tab[][]

        jns petlaM
        dec rdx;
    jnz petlaN
    ret
Ify2 endp
```

Zadanie 3 Wymyśl inne zadanie wykorzystujące instrukcje warunkowe na rejestrach wektorowych AVX 256bit i je zrealizuj. Forma dowolna X86/x64. Dane dowolne INT/Float/Double/INT64. Zbiór danych dowolne Tablica: jedno wymiarowa … n-wymiarowa.