Межведомственный суперкомпьютерный центр Российской академии наук –
филиал Федерального государственного учреждения «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН)

Бендерский Леонид Александрович
Рыбаков Алексей Анатольевич
Шумилин Сергей Сергеевич

Векторизация перемножения матриц специального вида с использованием инструкций AVX-512

III Международная научная конференция
«Конвергентные когнитивно-информационные технологии»
29.11.2018 – 02.12.2018, Московский Государственный Университет им. М. В. Ломоносова

Профиль исполнения расчетных кодов для численного решения задач газовой динамики

% time	Cumu-lative seconds	self seconds	calls	self s/call	total s/call	name
38.61	347.89	347.89	570077429	0.00	0.00	__mmatalg_MOD_mmult1
11.60	452.46	104.57	110	0.95	4.71	resd3d_
8.59	529.90	77.44	110	0.70	6.73	solv3d_
8.57	607.12	77.21	220	0.35	0.35	rstdsa_
6.12	662.28	55.16	799303681	0.00	0.00	__mmatalg_MOD_matvek2
5.15	708.69	46.41	114185753	0.00	0.00	gz3d3d1_
4.67	750.77	42.08	220	0.19	0.19	rstcinw_
2.62	774.38	23.61	84485163	0.00	0.00	__mmatalg_MOD_mmult
2.02	792.62	18.24	31111367	0.00	0.00	__mmatalg_MOD_mi0dnd
1.89	809.61	16.99	285038110	0.00	0.00	diamat1_
1.37	821.95	12.34	110	0.11	0.39	resd3b_
1.16	832.44	10.50	81821796	0.00	0.00	__mvtns3d_MOD_vtns3d
1.02	841.66	9.22	440	0.02	0.02	gszdtu_
1.00	850.68	9.02	404804228	0.00	0.00	weno5_
0.82	858.08	7.40	142518595	0.00	0.00	cflrim11_
0.70	864.35	6.27	71260947	0.00	0.00	__mmatalg_MOD_matvek1
0.58	869.58	5.23	330	0.02	0.02	corrsa_
0.58	874.81	5.23	220	0.02	0.02	rcbnsaw_
0.34	877.86	3.05	11	0.28	0.28	__module_blocks_MOD_tblock_activate
0.32	880.76	2.90	110	0.03	0.03	soursa_
0.30	883.43	2.67	330	0.01	0.01	main

Расположение матриц 8 x 8, 7 x 7, 6 x 6, 5 x 5 внутри охватывающей матрицы 8 x 8

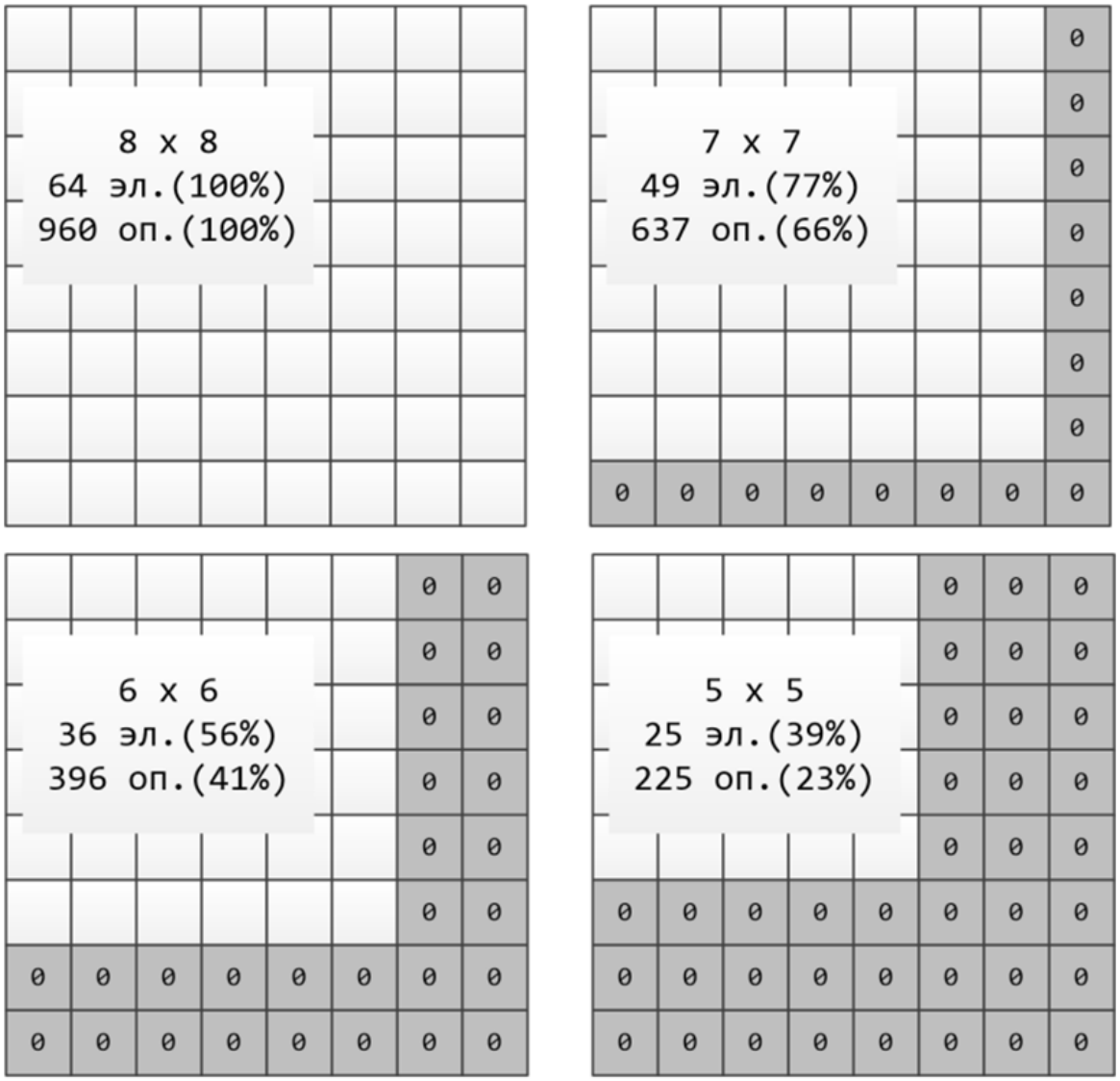


Схема выполнения перемножения двух квадратных матриц размера 8 x 8

```
01  void mul_8x8_orig(float * __restrict a,
02                    float * __restrict b,
03                    float * __restrict r)
04  {
05      for (int i = 0; i < V8; i++)
06      {
07          int ii = i * V8;
08
09          for (int j = 0; j < V8; j++)
10          {
11              float sum = 0.0;
12
13              for (int k = 0; k < V8; k++)
14              {
15                  int kk = k * V8;
16
17                  sum = sum + a[ii + k] * b[kk + j];
18              }
19
20              r[ii + j] = sum;
21          }
22      }
23  }
```

LIBXSMM What's inside?
March 5th, CINECA, Bologna, 2018, IXPUG.

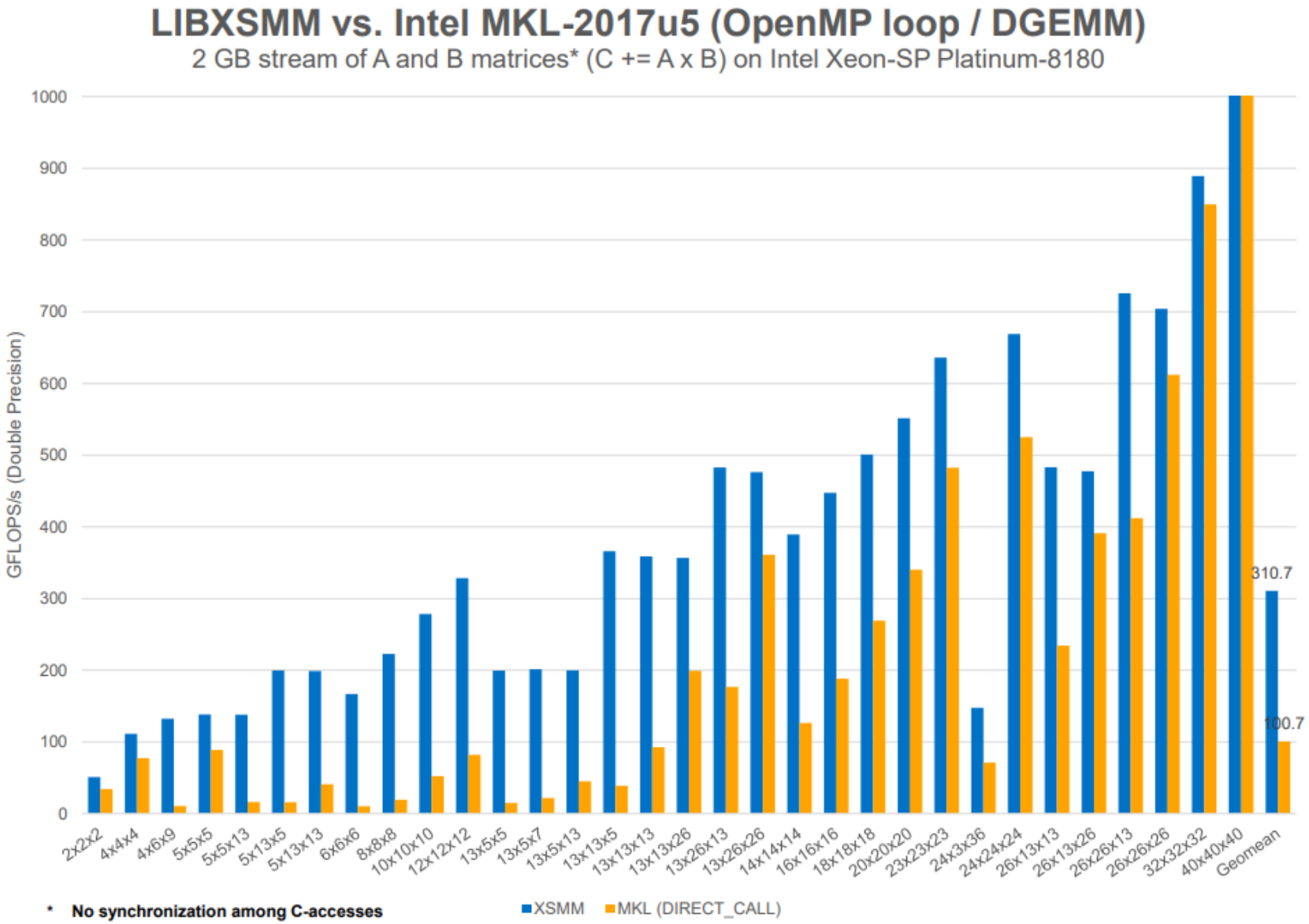


Иллюстрация работы простых двухоперандовых векторных инструкций из набора AVX-512

_mm512_add_ps	_mm512_mask_add_ps
_mm512_sub_ps	_mm512_mask_sub_ps
_mm512_mul_ps	_mm512_mask_mul_ps
_mm512_div_ps	_mm512_mask_div_ps
_mm512_pow_ps	_mm512_mask_pow_ps
_mm512_sqrt_ps	_mm512_mask_sqrt_ps

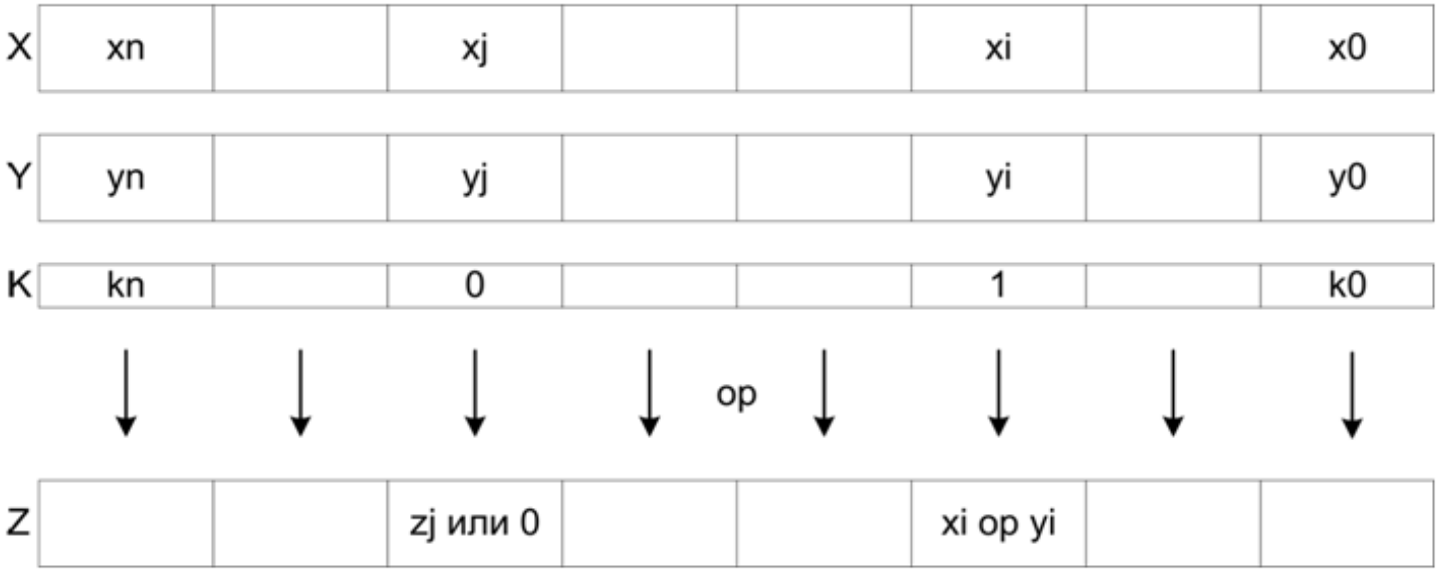


Иллюстрация работы FMA инструкций

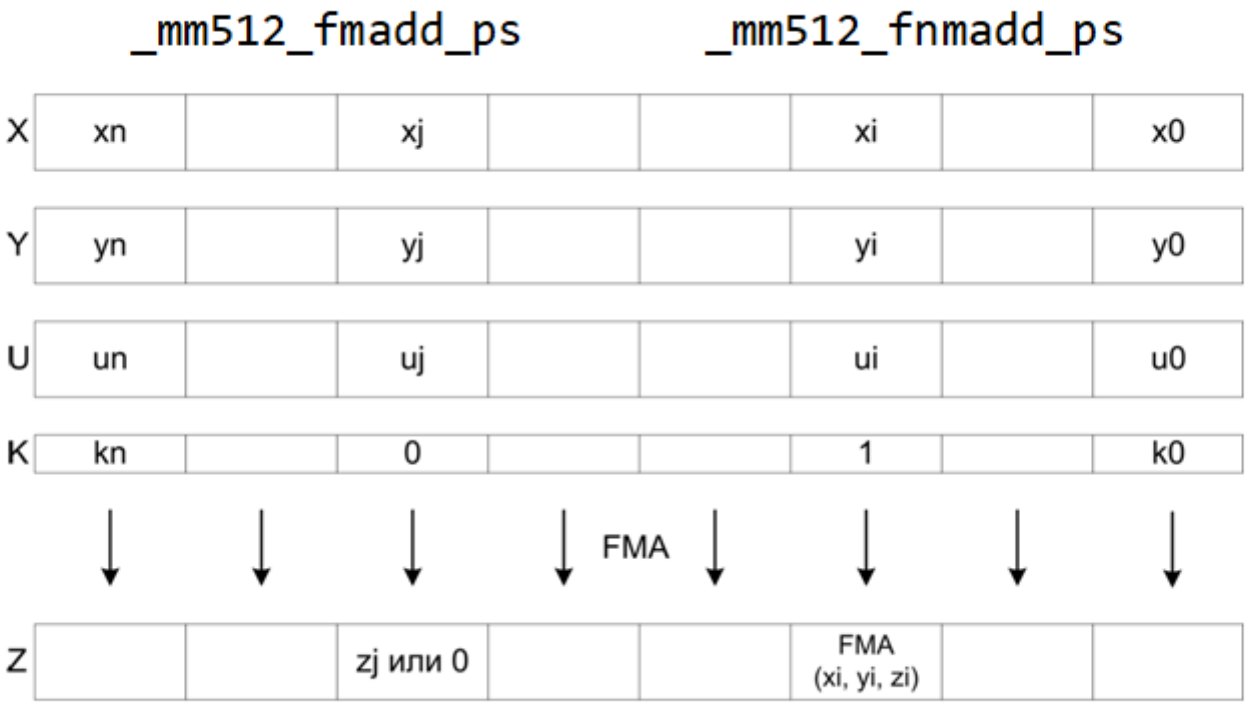


Иллюстрация работы векторных инструкций слияния по маске

```
VBLENDMPS (EVEX encoded versions)
(KL, VL) = (4, 128), (8, 256), (16, 512)
FOR j ← 0 TO KL-1
    i ← j * 32
    IF k1[j] OR *no controlmask*
        THEN
            IF (EVEX.b = 1) AND (SRC2 *is memory*)
                THEN
                    DEST[i+31:i] ← SRC2[31:0]
                ELSE
                    DEST[i+31:i] ← SRC2[i+31:i]
            FI;
        ELSE
            IF *merging-masking*                    ; merging-masking
                THEN DEST[i+31:i] ← SRC1[i+31:i]
                ELSE                                ; zeroing-masking
                    DEST[i+31:i] ← 0
            FI;
    FI;
ENDFOR
DEST[MAXVL-1:VL] ← 0
```

```
_mm512_mask_blend_ps
```

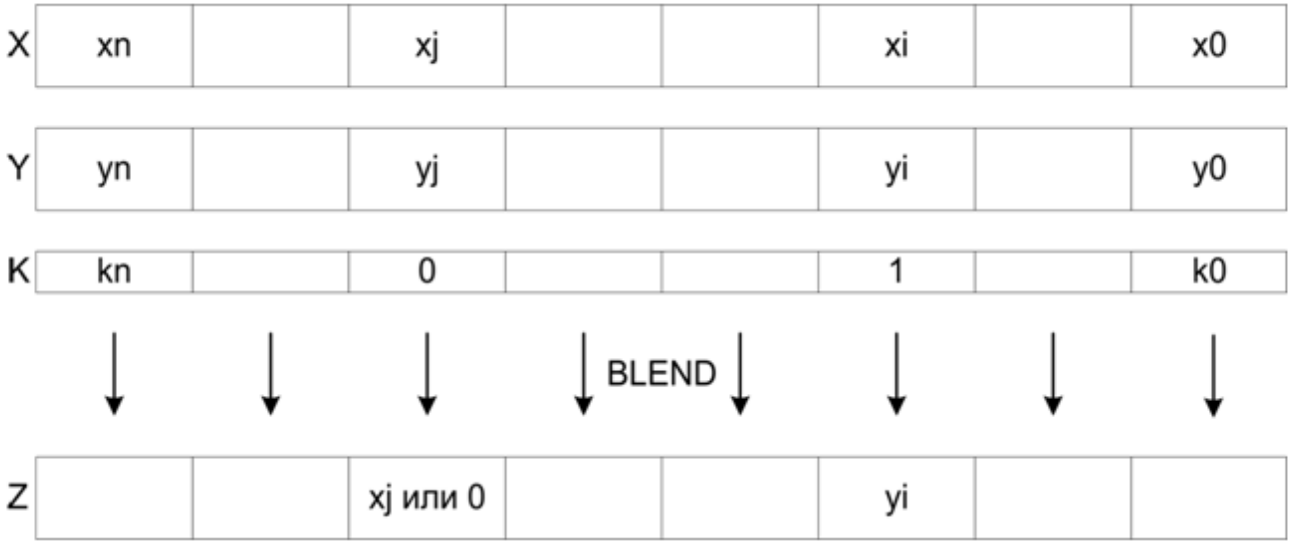


Иллюстрация работы различных инструкций перестановок элементов внутри векторов

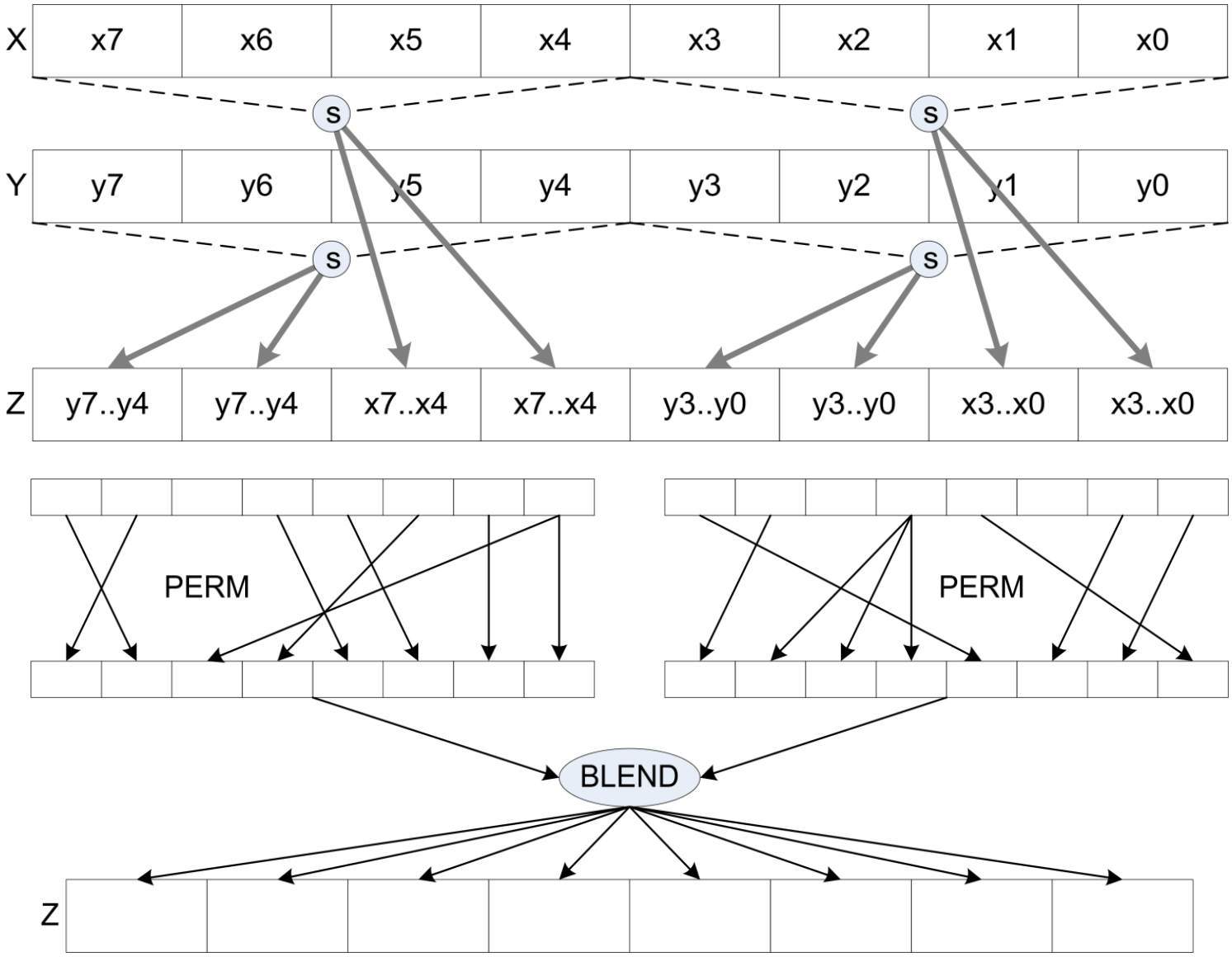


Иллюстрация работы операций множественного обращения в память

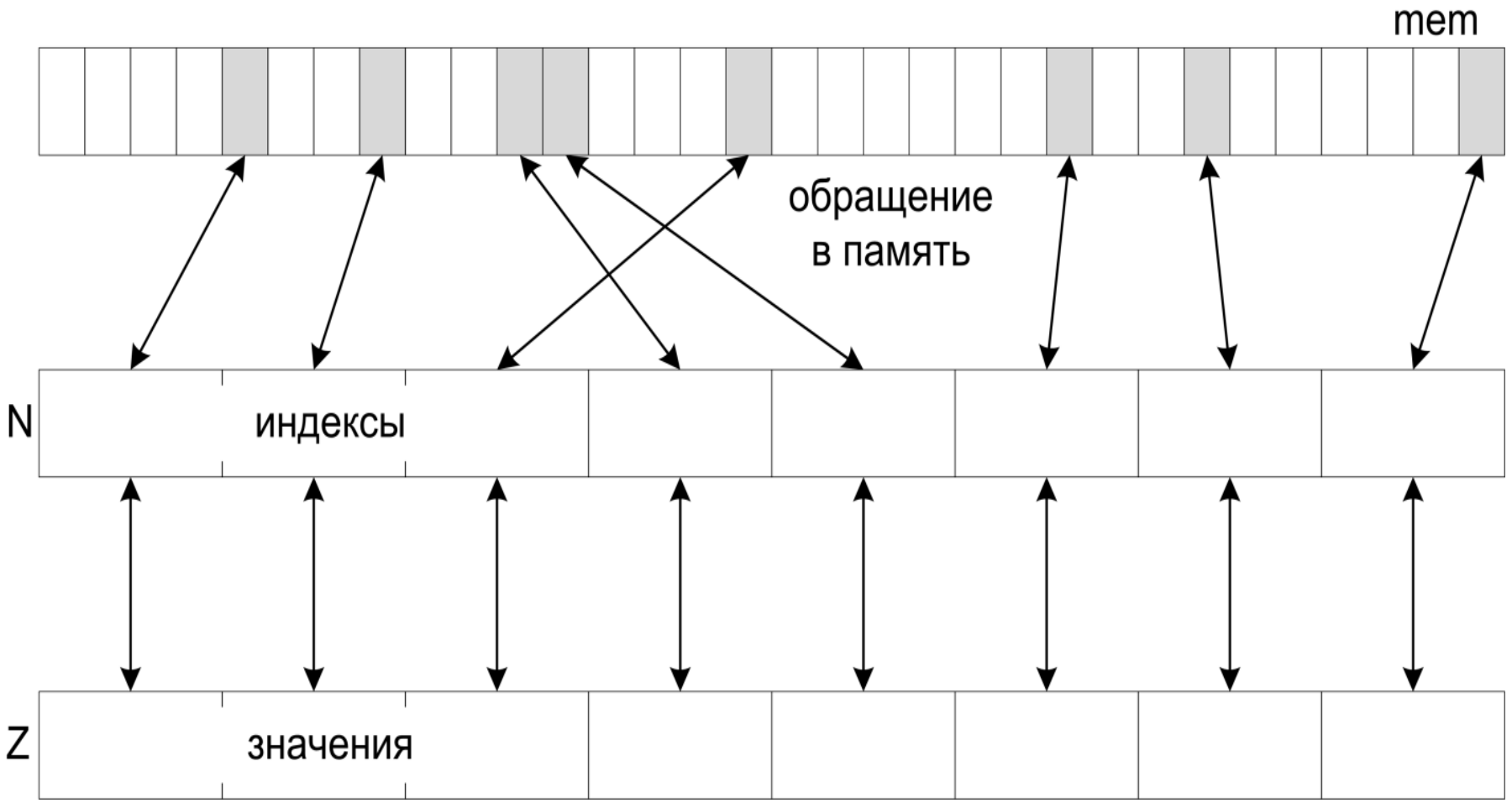


Схема реализации умножения матрицы на вектор

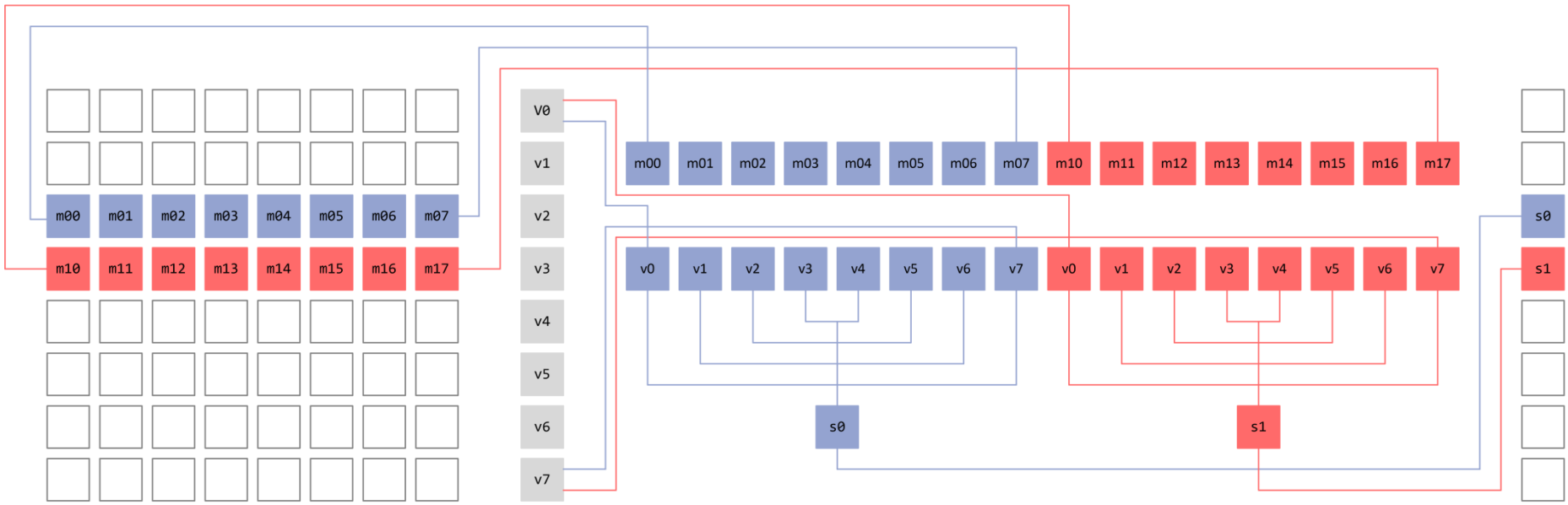
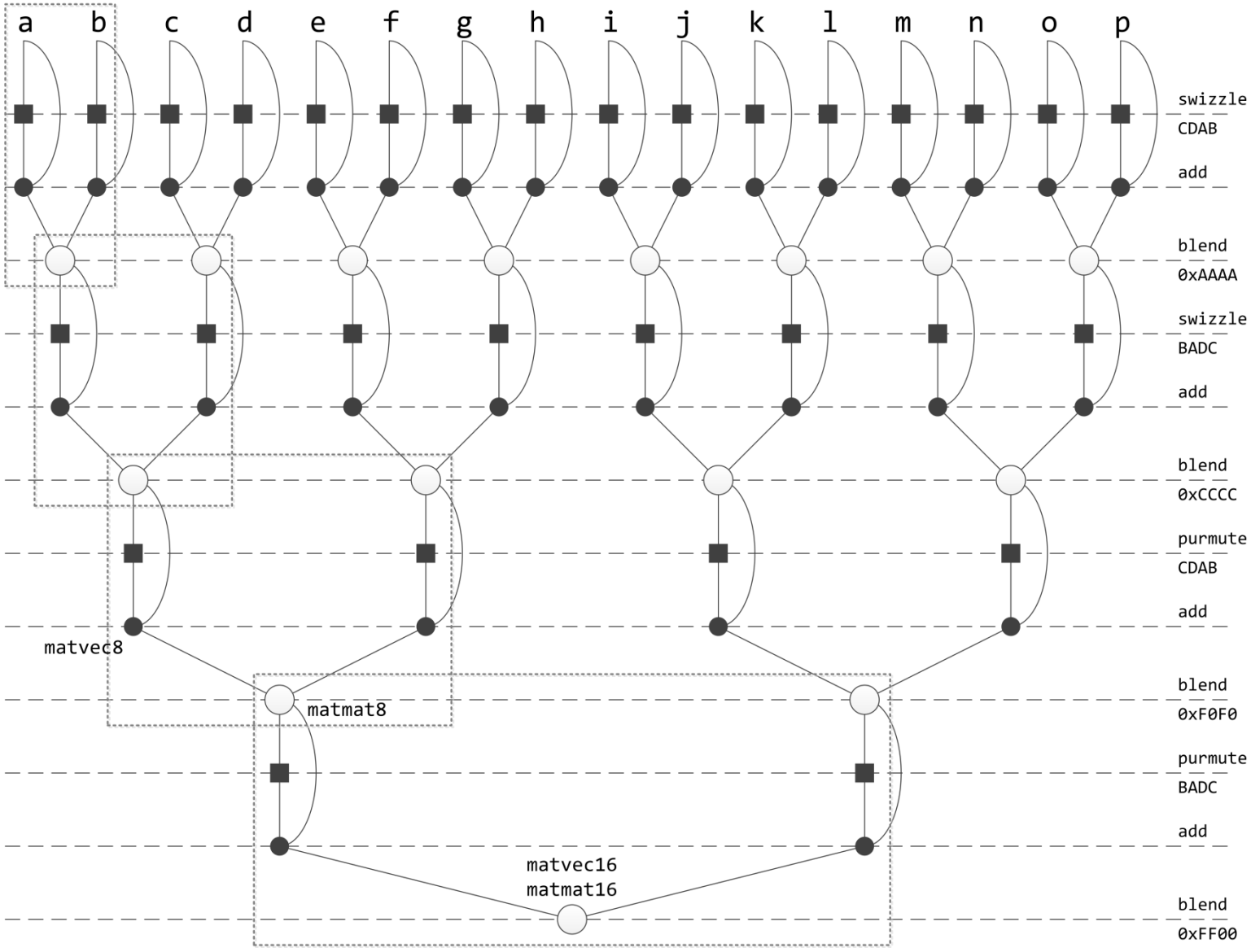
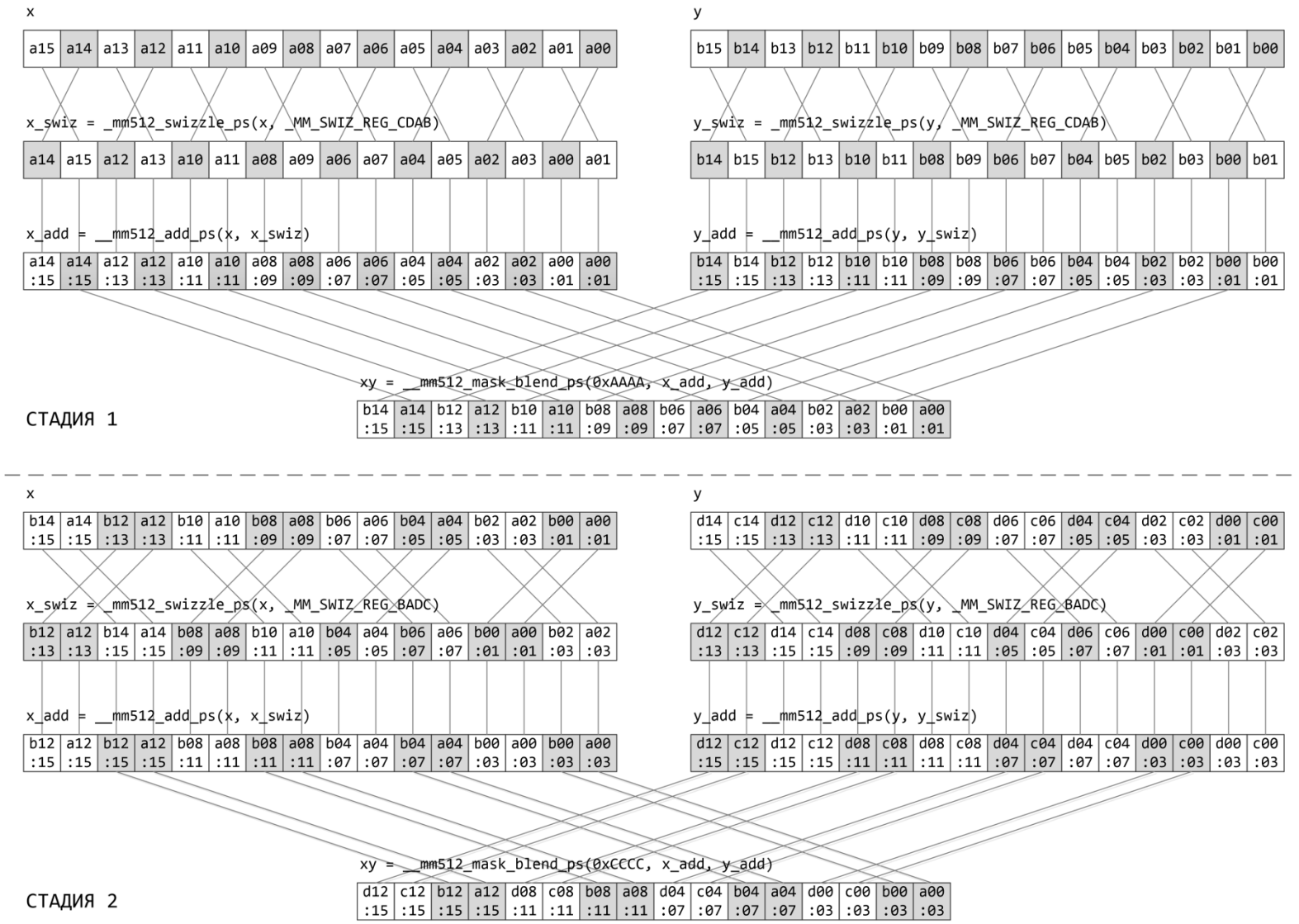


Схема реализации умножения матрицы на матрицу

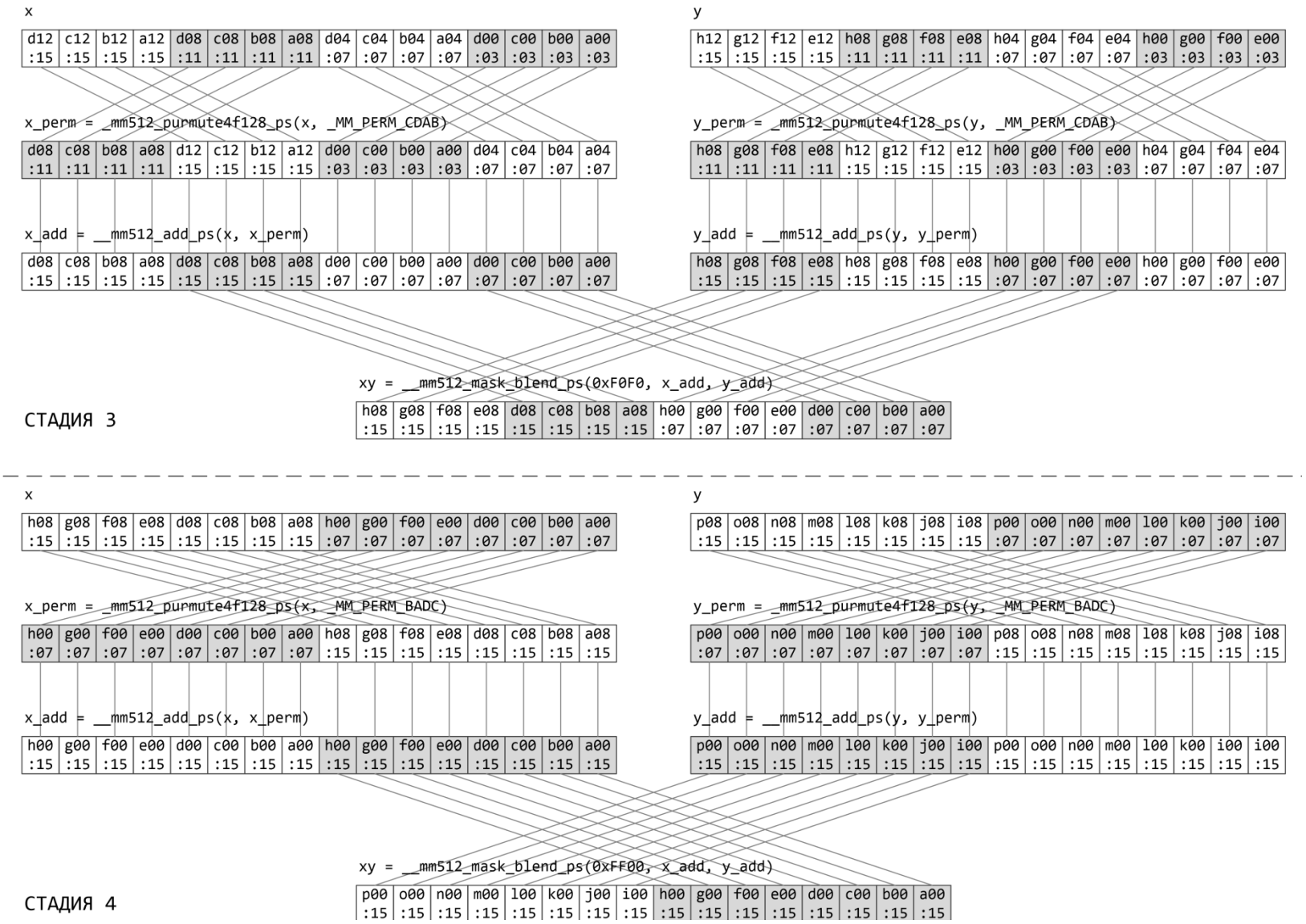
Дерево параллельного подсчета сумм элементов 16 различных векторов



Стадия 1 и Стадия 2 параллельного подсчета сумм элементов 16 векторов



Стадия 3 и Стадия 4 параллельного подсчета сумм элементов 16 векторов



Реализация связки в дереве параллельного вычисления сумм элементов векторов

```
01  #define SWIZ_2_ADD_2_BLEND_1(X, Y, SWIZ_TYPE, BLEND_MASK) \
02      _mm512_mask_blend_ps(BLEND_MASK, \
03                           ADD(X, \
04                               _mm512_swizzle_ps(X, \
05                                                 SWIZ_TYPE)), \
06                           ADD(Y, \
07                               _mm512_swizzle_ps(Y, \
08                                                 SWIZ_TYPE)))
09  #define PERM_2_ADD_2_BLEND_1(X, Y, PERM_TYPE, BLEND_MASK) \
10      _mm512_mask_blend_ps(BLEND_MASK, \
11                           ADD(X, \
12                               _mm512_permute4f128_ps(X, \
13                                                      PERM_TYPE)), \
14                           ADD(Y, \
15                               _mm512_permute4f128_ps(Y, \
16                                                      PERM_TYPE)))
```

Векторная реализация перемножения матриц с помощью параллельного подсчета сумм элементов векторов

```
01  void mul_8x8_opt(float * __restrict a, float * __restrict b, float * __restrict r)
02  {
03      ..........................................
04      __m512 bj, bj2,
05             m0, m1, m2, m3, m4, m5, m6, m7;
06      // Индексы для работы со столбцами.
07      __m512i ind_cc = _mm512_set_epi32(7 * V8 + 1, 6 * V8 + 1, 5 * V8 + 1, 4 * V8 + 1,
08                                        3 * V8 + 1, 2 * V8 + 1,     V8 + 1,          1,
09                                        7 * V8    , 6 * V8    , 5 * V8    , 4 * V8    ,
10                                        3 * V8    , 2 * V8    ,     V8    ,          0);
11      __m512i ind_st = _mm512_set_epi32(7 * V8    , 7 * V8 + 1, 5 * V8    , 5 * V8 + 1,
12                                        3 * V8    , 3 * V8 + 1,     V8    ,     V8 + 1,
13                                        6 * V8 + 1, 6 * V8    , 4 * V8 + 1, 4 * V8    ,
14                                        2 * V8 + 1, 2 * V8    ,     1     ,          0);
15      // Чтение всех строк матрицы a (по две в zmm регистр).
16      __m512 a0 = _mm512_load_ps(&a[0]);
17      ..........................................
18      __m512 a3 = _mm512_load_ps(&a[6 * V8]);
19
20      // Цикл по столбцам матрицы b.
21      for (int j = 0; j < V8; j += 2)
22      {
23          bj = _mm512_i32gather_ps(ind_cc, &b[j], _MM_SCALE_4);
24          bj2 = _mm512_permute4f128_ps(bj, _MM_PERM_BADC);
25          // Поэлементное перемножение строк a и столбцов b.
26          m0 = _mm512_mul_ps(a0, bj);
27          m1 = _mm512_mul_ps(a0, bj2);
28          ..........................................
29          m6 = _mm512_mul_ps(a3, bj);
30          m7 = _mm512_mul_ps(a3, bj2);
31          // Параллельное выполнение суммирования подвекторов.
32          m0 = SWIZ_2_ADD_2_BLEND_1(m0, m1, _MM_SWIZ_REG_CDAB, 0xAAAA);
33          m1 = SWIZ_2_ADD_2_BLEND_1(m2, m3, _MM_SWIZ_REG_CDAB, 0xAAAA);
34          m2 = SWIZ_2_ADD_2_BLEND_1(m4, m5, _MM_SWIZ_REG_CDAB, 0xAAAA);
35          m3 = SWIZ_2_ADD_2_BLEND_1(m6, m7, _MM_SWIZ_REG_CDAB, 0xAAAA);
36          m0 = SWIZ_2_ADD_2_BLEND_1(m0, m1, _MM_SWIZ_REG_BADC, 0xCCCC);
37          m1 = SWIZ_2_ADD_2_BLEND_1(m2, m3, _MM_SWIZ_REG_BADC, 0xCCCC);
38          m2 = PERM_2_ADD_2_BLEND_1(m0, m1, _MM_PERM_CDAB, 0xF0F0);
39          // Сохранение результата.
40          _mm512_i32scatter_ps(&r[j], ind_st, m2, _MM_SCALE_4);
41      }
42  }
```

Схема перемножения матриц в терминах операций со строками

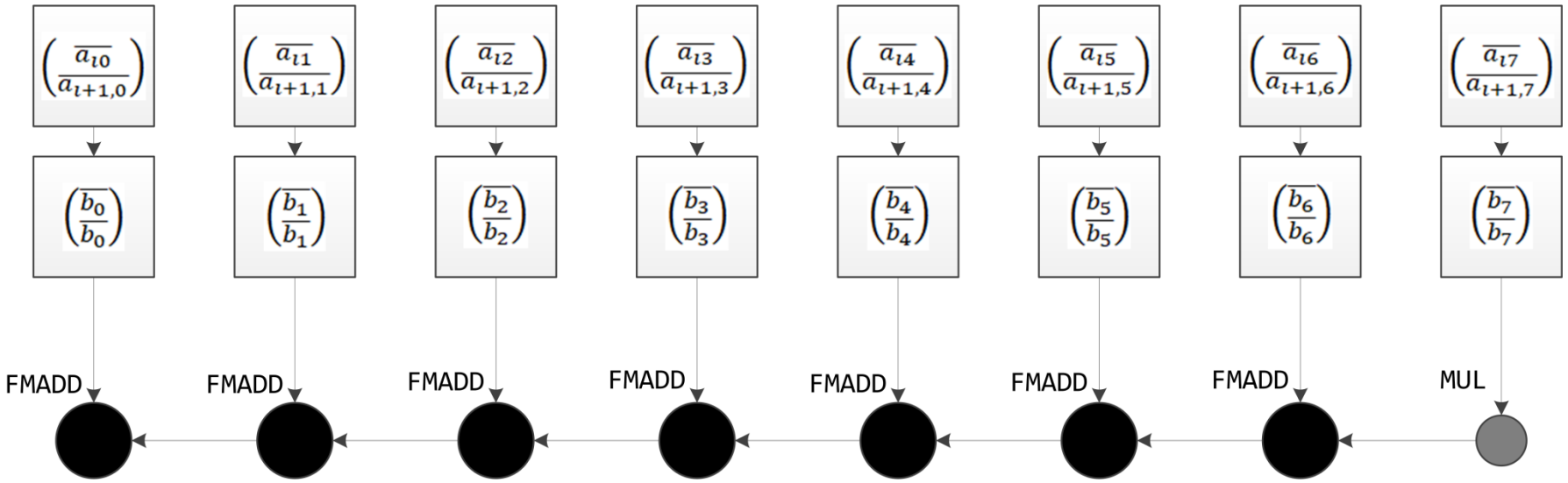
$$\begin{cases} r_{i0} = a_{i0}b_{00} + a_{i1}b_{10} + .. + a_{i7}b_{70} \\ .. \\ r_{i7} = a_{i0}b_{07} + a_{i1}b_{17} + .. + a_{i7}b_{77} \end{cases},$$

$$\begin{cases} r_{i+1,0} = a_{i+1,0}b_{00} + a_{i+1,1}b_{10} + .. + a_{i+1,7}b_{70} \\ .. \\ r_{+1,i7} = a_{i+1,0}b_{07} + a_{i+1,1}b_{17} + .. + a_{+1,i7}b_{77} \end{cases},$$

$$\overline{r_i} = a_{i0}\overline{b_0} + a_{i1}\overline{b_1} + .. + a_{i7}\overline{b_7}.$$

$$\overline{r_{i+1}} = a_{i+1,0}\overline{b_0} + a_{i+1,1}\overline{b_1} + .. + a_{i+1,7}\overline{b_7}.$$

$$\left(\frac{\overline{r_i}}{r_{i+1}}\right) = \left(\left(\frac{\overline{a_{i0}}}{a_{i+1,0}}\right) \circ \left(\frac{\overline{b_0}}{b_0}\right)\right) + \left(\left(\frac{\overline{a_{i1}}}{a_{i+1,1}}\right) \circ \left(\frac{\overline{b_1}}{b_1}\right)\right) + .. + \left(\left(\frac{\overline{a_{i7}}}{a_{i+1,7}}\right) \circ \left(\frac{\overline{b_7}}{b_7}\right)\right)$$



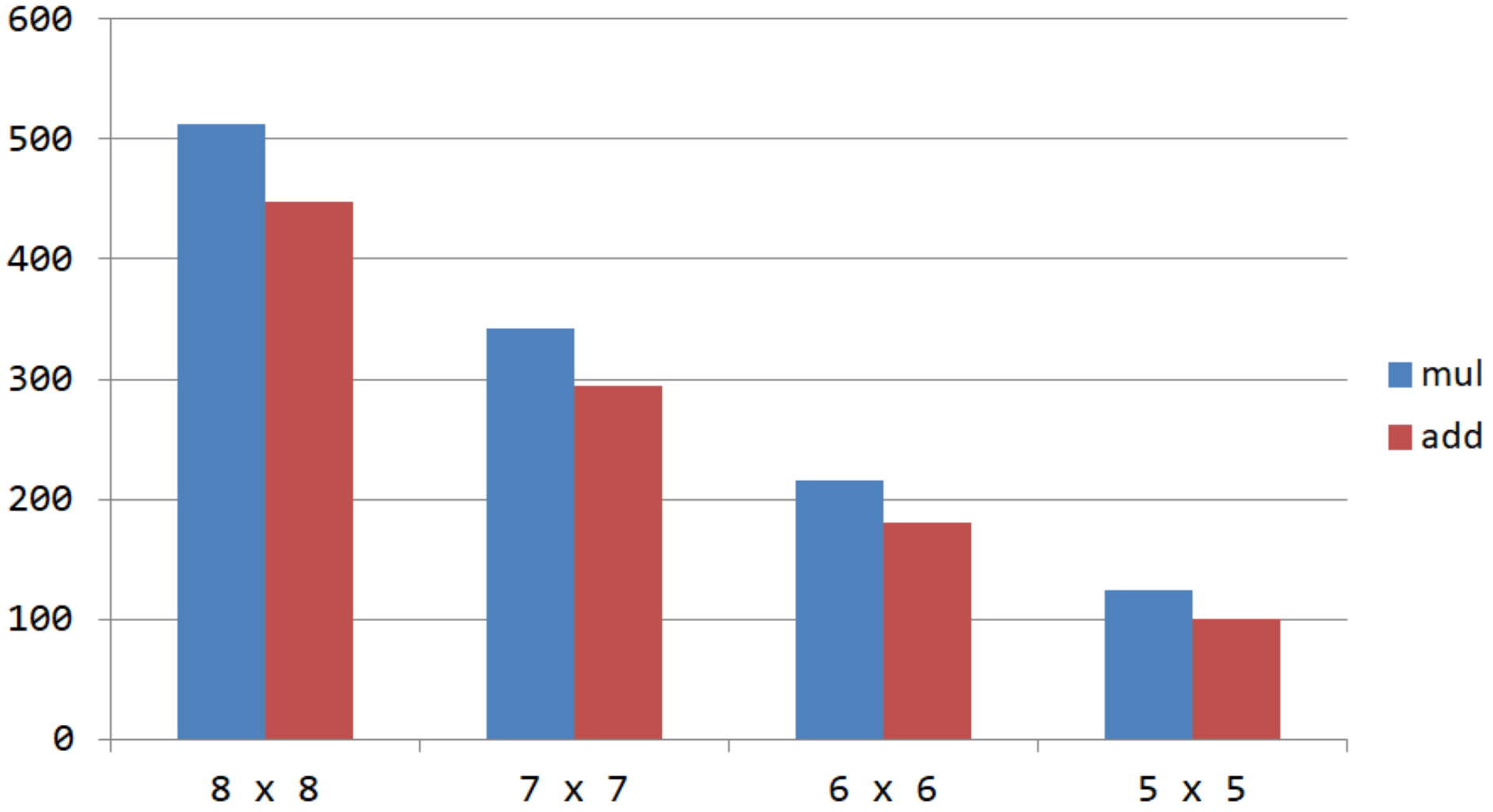
Векторная реализация перемножения матриц с использованием только операций со строками

```
01  void mul_8x8_opt(float * __restrict a, float * __b, float * __restrict r)
02  {
03      ....................................
04      // Индексы для дублирования первой и второй половины zmm регистра.
05      __m512i ind_df = _mm512_set_epi32( 7,  6,  5,  4,  3,  2, 1, 0,
06                                         7,  6,  5,  4,  3,  2, 1, 0);
07      __m512i ind_ds = _mm512_set_epi32(15, 14, 13, 12, 11, 10, 9, 8,
08                                        15, 14, 13, 12, 11, 10, 9, 8);
09      // Загрузка всех строк матрицы b (b0-b7) с дублированием.
10      __m512 b0 = _mm512_load_ps(&b[0]);
11      __m512 b1 = _mm512_permutexvar_ps(ind_ds, b0);
12      b0 = _mm512_permutexvar_ps(ind_df, b0);
13      ....................................
14      __m512 b6 = _mm512_load_ps(&b[6 * V8]);
15      __m512 b7 = _mm512_permutexvar_ps(ind_ds, b6);
16      b6 = _mm512_permutexvar_ps(ind_df, b6);
17      // Загрузка всех строк матрицы a (по две в zmm регистр).
18      __m512 a0 = _mm512_load_ps(&a[0]);
19      ....................................
20      __m512 a6 = _mm512_load_ps(&a[6 * V8]);
21      // Индексы для выбора элементов матрицы a.
22      __m512i ind_0 = _mm512_set_epi32(8, <8 раз> 8, 0, <8 раз> 0);
23      ....................................
24      __m512i ind_7 = _mm512_set_epi32(15, <8 раз> 15, 7, <8 раз> 7);
25
26  // Определение основного блока вычислений.
27  #define PERMXV _mm512_permutexvar_ps
28  #define MUL _mm512_mul_ps
29  #define FMADD _mm512_fmadd_ps
30  #define BLOCK(N, A)                          \
31      _mm512_store_ps(&r[N * V8],              \
32        FMADD(PERMXV(ind_0, A), b0,            \
33      ....................................
34                  FMADD(PERMXV(ind_6, A), b6, \
35                    MUL(PERMXV(ind_7, A), b7)))))))));
36
37      // Вычисление и сохранение результата.
38      BLOCK(0, a0);
39      ....................................
40      BLOCK(6, a6);
41
42  #undef PERMXVAR
43  #undef MUL
44  #undef FMADD
45  #undef BLOCK
46
47  }
```

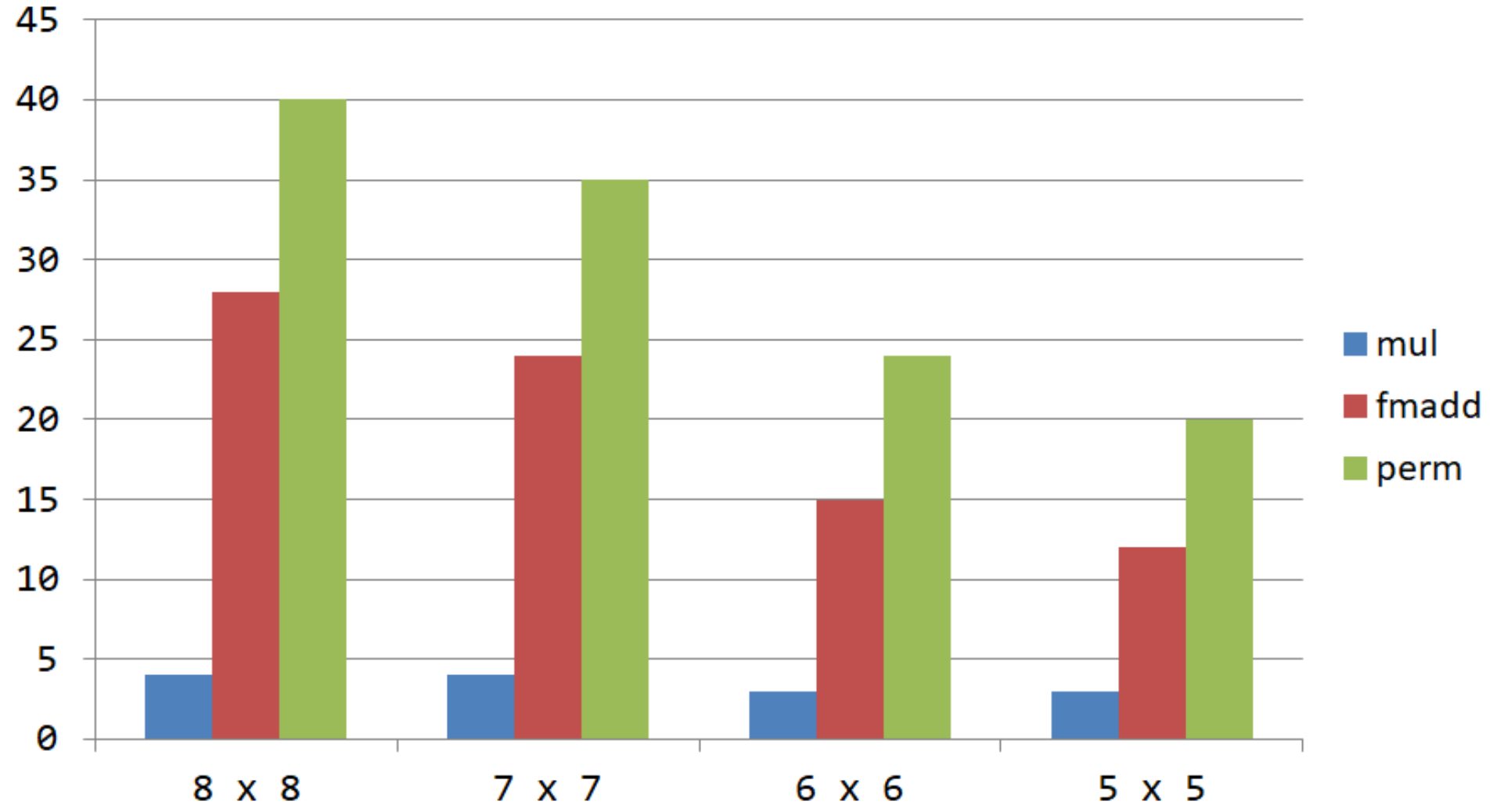
Ассемблерный код функции перемножения двух матриц (второй подход)

```
vmovups 0x152326(%rip),%zmm1

vmovups 0x15249c(%rip),%zmm18

vpermps 0x80(%rdx),%zmm1,%zmm19
vpermps (%rsi),%zmm18,%zmm2
vpermps 0x40(%rsi),%zmm18,%zmm8
vpermps 0x80(%rsi),%zmm18,%zmm20
vmulps %zmm19,%zmm2,%zmm3
vmovups 0x152331(%rip),%zmm0

vmovups 0x152427(%rip),%zmm17

vmulps %zmm8,%zmm19,%zmm9
vmulps %zmm20,%zmm19,%zmm22
vpermps 0x40(%rdx),%zmm0,%zmm21
vpermps (%rsi),%zmm17,%zmm4
vpermps 0x40(%rsi),%zmm17,%zmm10
vpermps 0x80(%rsi),%zmm17,%zmm24
vfmadd213ps %zmm3,%zmm21,%zmm4
vmovups 0x1523b0(%rip),%zmm16

vfmadd213ps %zmm9,%zmm21,%zmm10
vpermps 0x40(%rdx),%zmm1,%zmm23
vmovups 0x152359(%rip),%zmm15

vfmadd213ps %zmm22,%zmm21,%zmm24
vpermps (%rsi),%zmm16,%zmm5
vpermps 0x40(%rsi),%zmm16,%zmm11
vpermps 0x80(%rsi),%zmm16,%zmm26
vfmadd213ps %zmm4,%zmm23,%zmm5
vfmadd213ps %zmm10,%zmm23,%zmm11
vpermps 0x40(%rsi),%zmm15,%zmm12
vpermps (%rsi),%zmm15,%zmm6
vpermps (%rdx),%zmm0,%zmm25
vpermps 0x80(%rsi),%zmm15,%zmm28
vfmadd213ps %zmm24,%zmm23,%zmm26
vmovups 0x1522c9(%rip),%zmm14

vpermps (%rdx),%zmm1,%zmm27
vfmadd213ps %zmm5,%zmm25,%zmm6
vfmadd213ps %zmm11,%zmm25,%zmm12
vpermps (%rsi),%zmm14,%zmm7
vpermps 0x40(%rsi),%zmm14,%zmm13
vfmadd213ps %zmm26,%zmm25,%zmm28
vpermps 0x80(%rsi),%zmm14,%zmm29
vfmadd213ps %zmm6,%zmm27,%zmm7
vmovups %zmm7,(%rdi)
vfmadd213ps %zmm12,%zmm27,%zmm13
vmovups %zmm13,0x40(%rdi)
vfmadd213ps %zmm28,%zmm27,%zmm29
vmovups %zmm29,0x80(%rdi)
retq
nopw   0x0(%rax,%rax,1)
```

Количество операций умножения и сложения для реализации перемножения двух матриц



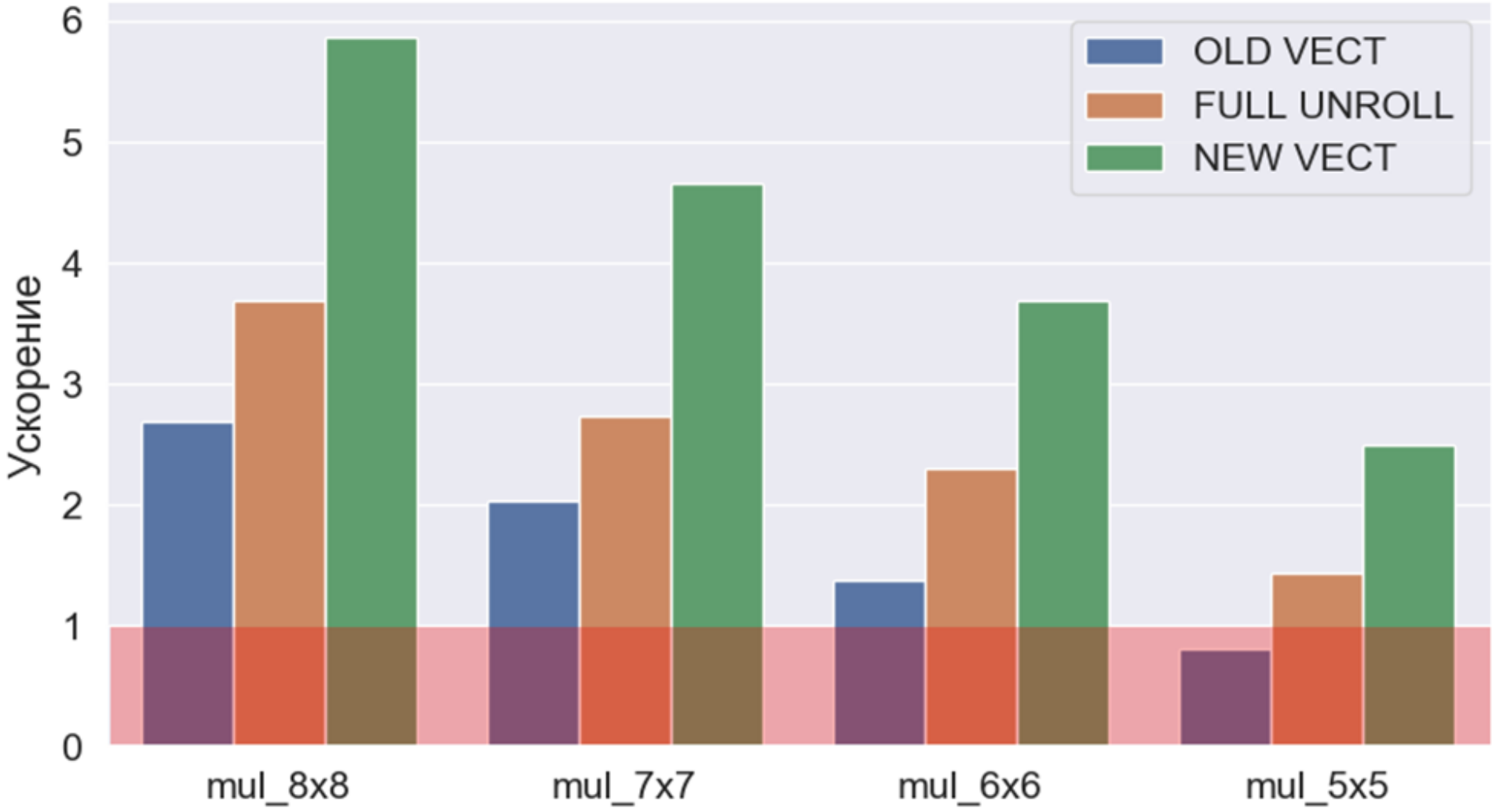
Количество векторных операций mul, fmadd и perm для реализации перемножения матриц



Прямое вычисление всех элементов результирующей матрицы

```
01    r[ADR(0, 0)] = a[ADR(0,0)]*b[ADR(0,0)]+a[ADR(0,1)]*b[ADR(1,0)]+a[ADR(0,2)]*b[ADR(2,0)]+a[ADR(0,3)]*b[ADR(3,0)]
02                 + a[ADR(0,4)]*b[ADR(4,0)]+a[ADR(0,5)]*b[ADR(5,0)]+a[ADR(0,6)]*b[ADR(6,0)]+a[ADR(0,7)]*b[ADR(7,0)];
03    r[ADR(0, 1)] = a[ADR(0,0)]*b[ADR(0,1)]+a[ADR(0,1)]*b[ADR(1,1)]+a[ADR(0,2)]*b[ADR(2,1)]+a[ADR(0,3)]*b[ADR(3,1)]
04                 + a[ADR(0,4)]*b[ADR(4,1)]+a[ADR(0,5)]*b[ADR(5,1)]+a[ADR(0,6)]*b[ADR(6,1)]+a[ADR(0,7)]*b[ADR(7,1)];
05    r[ADR(0, 2)] = a[ADR(0,0)]*b[ADR(0,2)]+a[ADR(0,1)]*b[ADR(1,2)]+a[ADR(0,2)]*b[ADR(2,2)]+a[ADR(0,3)]*b[ADR(3,2)]
06                 + a[ADR(0,4)]*b[ADR(4,2)]+a[ADR(0,5)]*b[ADR(5,2)]+a[ADR(0,6)]*b[ADR(6,2)]+a[ADR(0,7)]*b[ADR(7,2)];
07    r[ADR(0, 3)] = a[ADR(0,0)]*b[ADR(0,3)]+a[ADR(0,1)]*b[ADR(1,3)]+a[ADR(0,2)]*b[ADR(2,3)]+a[ADR(0,3)]*b[ADR(3,3)]
08                 + a[ADR(0,4)]*b[ADR(4,3)]+a[ADR(0,5)]*b[ADR(5,3)]+a[ADR(0,6)]*b[ADR(6,3)]+a[ADR(0,7)]*b[ADR(7,3)];
09    r[ADR(0, 4)] = a[ADR(0,0)]*b[ADR(0,4)]+a[ADR(0,1)]*b[ADR(1,4)]+a[ADR(0,2)]*b[ADR(2,4)]+a[ADR(0,3)]*b[ADR(3,4)]
10                 + a[ADR(0,4)]*b[ADR(4,4)]+a[ADR(0,5)]*b[ADR(5,4)]+a[ADR(0,6)]*b[ADR(6,4)]+a[ADR(0,7)]*b[ADR(7,4)];
11    r[ADR(0, 5)] = a[ADR(0,0)]*b[ADR(0,5)]+a[ADR(0,1)]*b[ADR(1,5)]+a[ADR(0,2)]*b[ADR(2,5)]+a[ADR(0,3)]*b[ADR(3,5)]
12                 + a[ADR(0,4)]*b[ADR(4,5)]+a[ADR(0,5)]*b[ADR(5,5)]+a[ADR(0,6)]*b[ADR(6,5)]+a[ADR(0,7)]*b[ADR(7,5)];
13    r[ADR(0, 6)] = a[ADR(0,0)]*b[ADR(0,6)]+a[ADR(0,1)]*b[ADR(1,6)]+a[ADR(0,2)]*b[ADR(2,6)]+a[ADR(0,3)]*b[ADR(3,6)]
14                 + a[ADR(0,4)]*b[ADR(4,6)]+a[ADR(0,5)]*b[ADR(5,6)]+a[ADR(0,6)]*b[ADR(6,6)]+a[ADR(0,7)]*b[ADR(7,6)];
15    r[ADR(0, 7)] = a[ADR(0,0)]*b[ADR(0,7)]+a[ADR(0,1)]*b[ADR(1,7)]+a[ADR(0,2)]*b[ADR(2,7)]+a[ADR(0,3)]*b[ADR(3,7)]
16                 + a[ADR(0,4)]*b[ADR(4,7)]+a[ADR(0,5)]*b[ADR(5,7)]+a[ADR(0,6)]*b[ADR(6,7)]+a[ADR(0,7)]*b[ADR(7,7)];
17
18    r[ADR(1, 0)] = a[ADR(1,0)]*b[ADR(0,0)]+a[ADR(1,1)]*b[ADR(1,0)]+a[ADR(1,2)]*b[ADR(2,0)]+a[ADR(1,3)]*b[ADR(3,0)]
19                 + a[ADR(1,4)]*b[ADR(4,0)]+a[ADR(1,5)]*b[ADR(5,0)]+a[ADR(1,6)]*b[ADR(6,0)]+a[ADR(1,7)]*b[ADR(7,0)];
20    r[ADR(1, 1)] = a[ADR(1,0)]*b[ADR(0,1)]+a[ADR(1,1)]*b[ADR(1,1)]+a[ADR(1,2)]*b[ADR(2,1)]+a[ADR(1,3)]*b[ADR(3,1)]
21                 + a[ADR(1,4)]*b[ADR(4,1)]+a[ADR(1,5)]*b[ADR(5,1)]+a[ADR(1,6)]*b[ADR(6,1)]+a[ADR(1,7)]*b[ADR(7,1)];
22    r[ADR(1, 2)] = a[ADR(1,0)]*b[ADR(0,2)]+a[ADR(1,1)]*b[ADR(1,2)]+a[ADR(1,2)]*b[ADR(2,2)]+a[ADR(1,3)]*b[ADR(3,2)]
23                 + a[ADR(1,4)]*b[ADR(4,2)]+a[ADR(1,5)]*b[ADR(5,2)]+a[ADR(1,6)]*b[ADR(6,2)]+a[ADR(1,7)]*b[ADR(7,2)];
24    r[ADR(1, 3)] = a[ADR(1,0)]*b[ADR(0,3)]+a[ADR(1,1)]*b[ADR(1,3)]+a[ADR(1,2)]*b[ADR(2,3)]+a[ADR(1,3)]*b[ADR(3,3)]
25                 + a[ADR(1,4)]*b[ADR(4,3)]+a[ADR(1,5)]*b[ADR(5,3)]+a[ADR(1,6)]*b[ADR(6,3)]+a[ADR(1,7)]*b[ADR(7,3)];
26    r[ADR(1, 4)] = a[ADR(1,0)]*b[ADR(0,4)]+a[ADR(1,1)]*b[ADR(1,4)]+a[ADR(1,2)]*b[ADR(2,4)]+a[ADR(1,3)]*b[ADR(3,4)]
27                 + a[ADR(1,4)]*b[ADR(4,4)]+a[ADR(1,5)]*b[ADR(5,4)]+a[ADR(1,6)]*b[ADR(6,4)]+a[ADR(1,7)]*b[ADR(7,4)];
28    r[ADR(1, 5)] = a[ADR(1,0)]*b[ADR(0,5)]+a[ADR(1,1)]*b[ADR(1,5)]+a[ADR(1,2)]*b[ADR(2,5)]+a[ADR(1,3)]*b[ADR(3,5)]
29                 + a[ADR(1,4)]*b[ADR(4,5)]+a[ADR(1,5)]*b[ADR(5,5)]+a[ADR(1,6)]*b[ADR(6,5)]+a[ADR(1,7)]*b[ADR(7,5)];
30    r[ADR(1, 6)] = a[ADR(1,0)]*b[ADR(0,6)]+a[ADR(1,1)]*b[ADR(1,6)]+a[ADR(1,2)]*b[ADR(2,6)]+a[ADR(1,3)]*b[ADR(3,6)]
31                 + a[ADR(1,4)]*b[ADR(4,6)]+a[ADR(1,5)]*b[ADR(5,6)]+a[ADR(1,6)]*b[ADR(6,6)]+a[ADR(1,7)]*b[ADR(7,6)];
32    r[ADR(1, 7)] = a[ADR(1,0)]*b[ADR(0,7)]+a[ADR(1,1)]*b[ADR(1,7)]+a[ADR(1,2)]*b[ADR(2,7)]+a[ADR(1,3)]*b[ADR(3,7)]
33                 + a[ADR(1,4)]*b[ADR(4,7)]+a[ADR(1,5)]*b[ADR(5,7)]+a[ADR(1,6)]*b[ADR(6,7)]+a[ADR(1,7)]*b[ADR(7,7)];
```

Ускорение, достигаемое при использовании трех разных подходов к перемножению матриц



Спасибо за внимание!

Бендерский Леонид Александрович
leosun.ben@gmail.com

Рыбаков Алексей Анатольевич
rybakov@jscc.ru

Шумилин Сергей Сергеевич
shumilin@jscc.ru