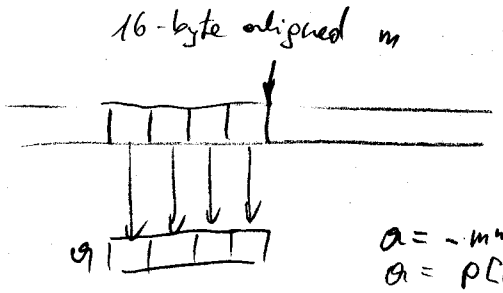
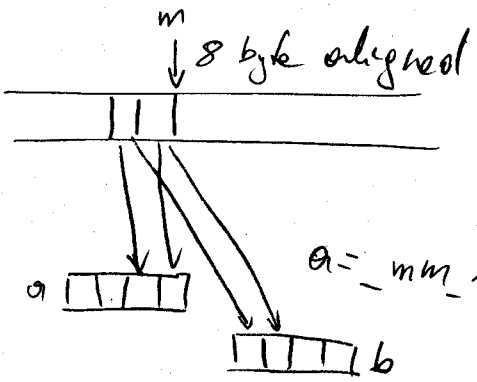
Table of Contents

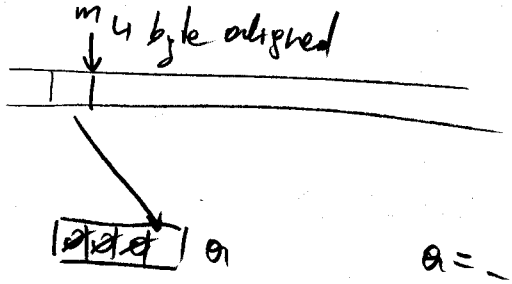
Load Instructions




```
a = _mm_load_ps(m);      aligned, explicit
a = p[i];                 - " -, implicit
a = _mm_loadu_ps(m);      unaligned
```




```
a = _mm_loadl_pi(a, m)
b = _mm_loadh_pi(b, m)
```




```
a = _mm_load_ss(m)
```

Constants (Special Load instructions)

```
C = _mm_set_ps(1.0, 2.0, 3.0, 4.0);
o1 = _mm_set1_ps(1.0);
e = _mm_set_ss(1.0);
f = _mm_setzero_ps();
```

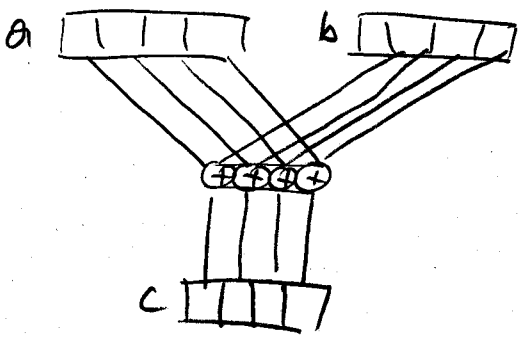
4.0	3.0	2.0	1.0

1.0	1.0	1.0	1.0

0	0	0	1.0

0	0	0	0

Vector arithmetic

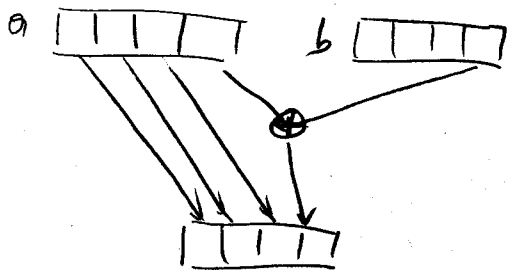



```
c = _mm_addps(a, b)
```

Some:

```
c = _mm_sub_ps(a,b)
c = _mm_mul_ps(a,b)
   ⋮
```

Scalar arithmetic

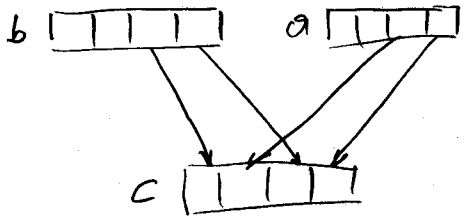



```
c = _mm_addss(a,b)
```

Reorder Instructions

Unpack lo

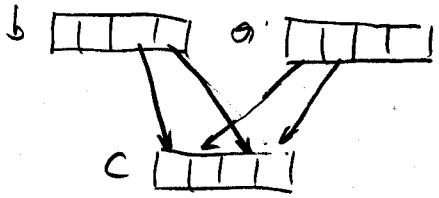
$c = \_mm\_unpacklo\_ps(a,b)$



$c_0 = a_0$
$c_1 = b_0$
$c_2 = a_1$
$c_3 = b_1$

Unpack hi

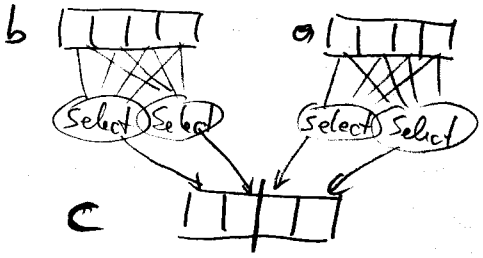
$c = \_mm\_unpackhi\_ps(a,b)$



$c_0 = a_2$
$c_1 = b_2$
$c_2 = a_3$
$c_3 = b_3$

Shuffle

$c = \_mm\_shuffle\_ps(a,b, \_MM\_SHUFFLE(l,k,j,i))$

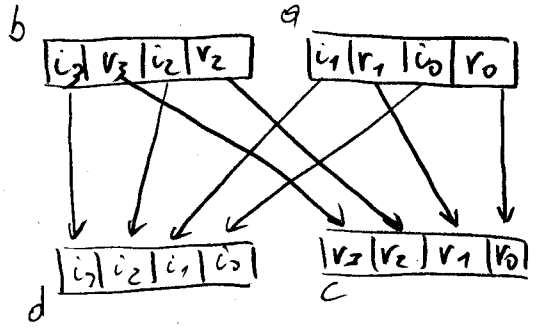


$c_0 = a_i$
$c_1 = a_j$
$c_2 = b_k$
$c_3 = b_l$

$i,j,k,l \in \{0,\ldots,3\}$
immediate

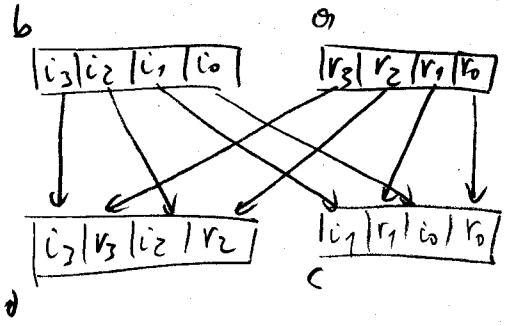
Reorder Instructions – Examples

Interleaved Complex → split complex : L_2^8



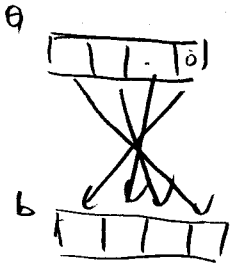

```
c = _mm_shuffle_ps(a, b, _MM_SHUFFLE(2,0,2,0));
d = _mm_shuffle_ps(a, b, _MM_SHUFFLE(3,1,3,1));
```

Split complex → interleaved complex : L_4^8




```
c = _mm_unpacklo_ps(a, b);
d = _mm_unpackhi_ps(a, b);
```

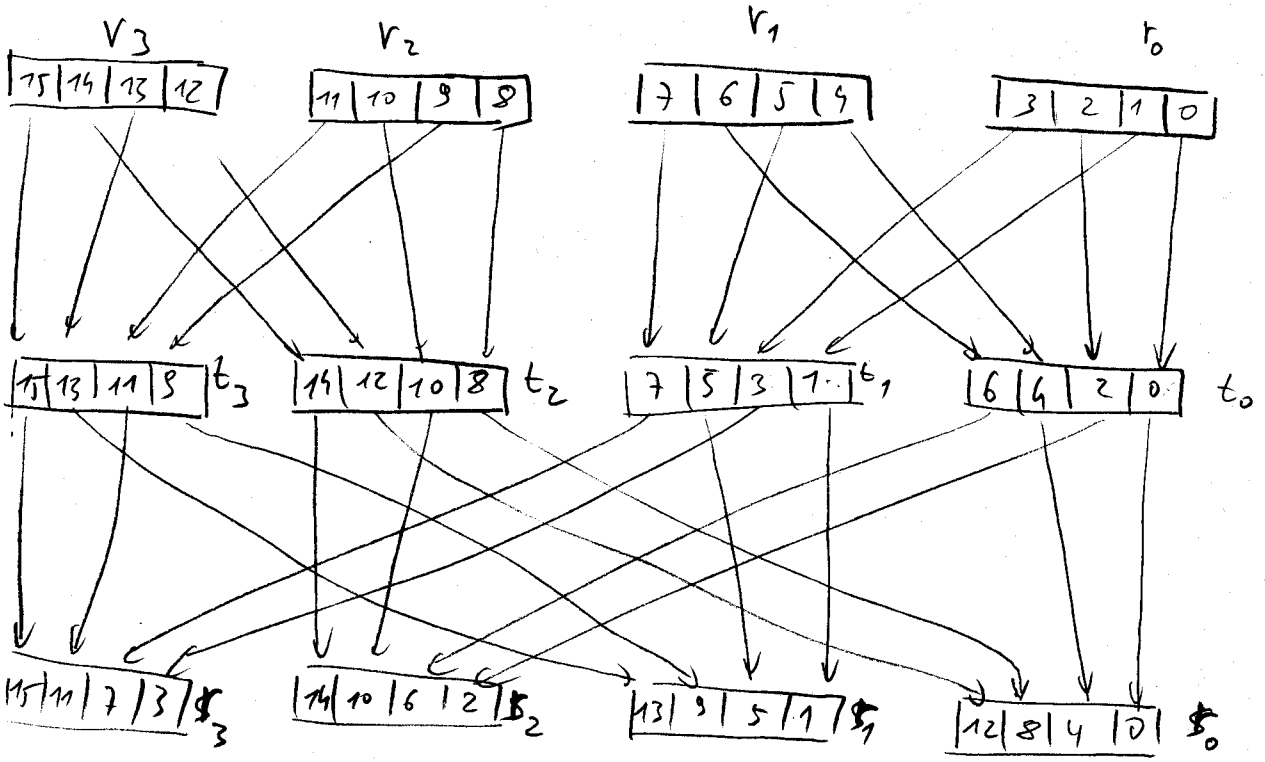
Reverse Vector : J_4




```
b = _mm_shuffle_ps(a, a, _MM_SHUFFLE(0,1,2,3));
```

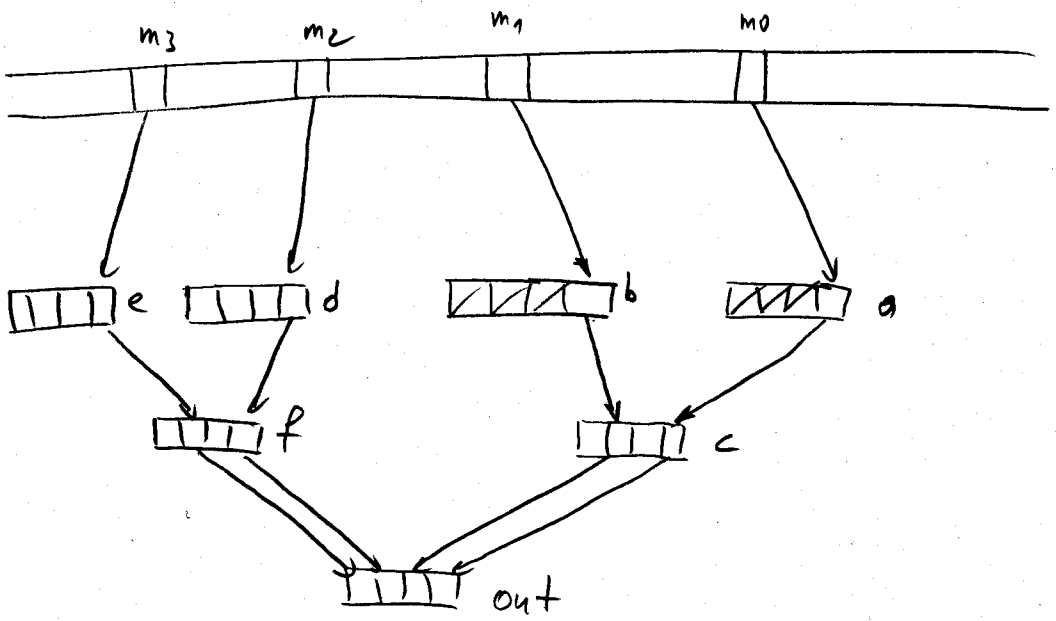
Reorder Instructions - Examples

Transpose 4x4 : $L^{16}_4 = \left(L^4_2 \otimes I_4\right)\left(I_2 \otimes L^8_2\right)\left(L^4_2 \otimes I_4\right)\left(I_2 \otimes L^8_2\right)$



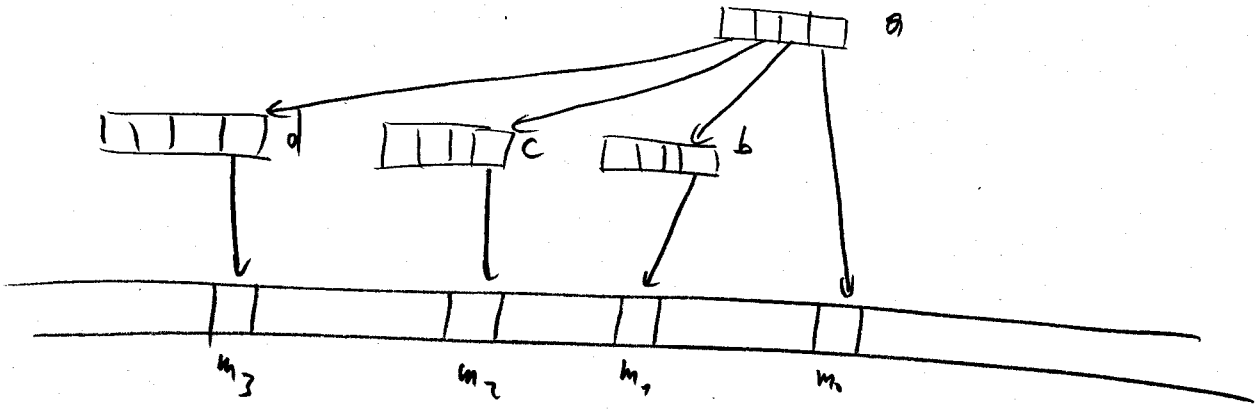

```
#define _MM_TRANSPOSE4_PS(r0, r1, r2, r3) { \
  __m128 t0, t1, t2, t3; \
  t0 = _mm_shuffle_ps(r0, r1, _MM_SHUFFLE(2, 0, 2, 0)); \
  t1 = _mm_shuffle_ps(r0, r1, _MM_SHUFFLE(3, 1, 3, 1)); \
  t2 = _mm_shuffle_ps(r2, r3, _MM_SHUFFLE(2, 0, 2, 0)); \
  t3 = _mm_shuffle_ps(r2, r3, _MM_SHUFFLE(3, 1, 3, 1)); \
  r0 = _mm_shuffle_ps(t0, t2, _MM_SHUFFLE(2, 0, 2, 0)); \
  r1 = _mm_shuffle_ps(t1, t3, _MM_SHUFFLE(2, 0, 2, 0)); \
  r2 = _mm_shuffle_ps(t0, t2, _MM_SHUFFLE(3, 1, 3, 1)); \
  r3 = _mm_shuffle_ps(t1, t3, _MM_SHUFFLE(3, 1, 3, 1)); \
}
```

Example: load 4 real numbers




```
#define SCALAR_LOAD (out, m0, m1, m2, m3) {\
    a = _mm_load_ss(m0);                  \
    b = _mm_load_ss(m1);                  \
    c = _mm_shuffle_ps(a, b, _MM_SHUFFLE(1,0,1,0));\
    d = _mm_load_ss(m2);                  \
    e = _mm_load_ss(m3);                  \
    f = _mm_shuffle_ps(d, e, _MM_SHUFFLE(1,0,1,0));\
    out = _mm_shuffle_ps(c, f, _MM_SHUFFLE(1,0,1,0));\
}
```

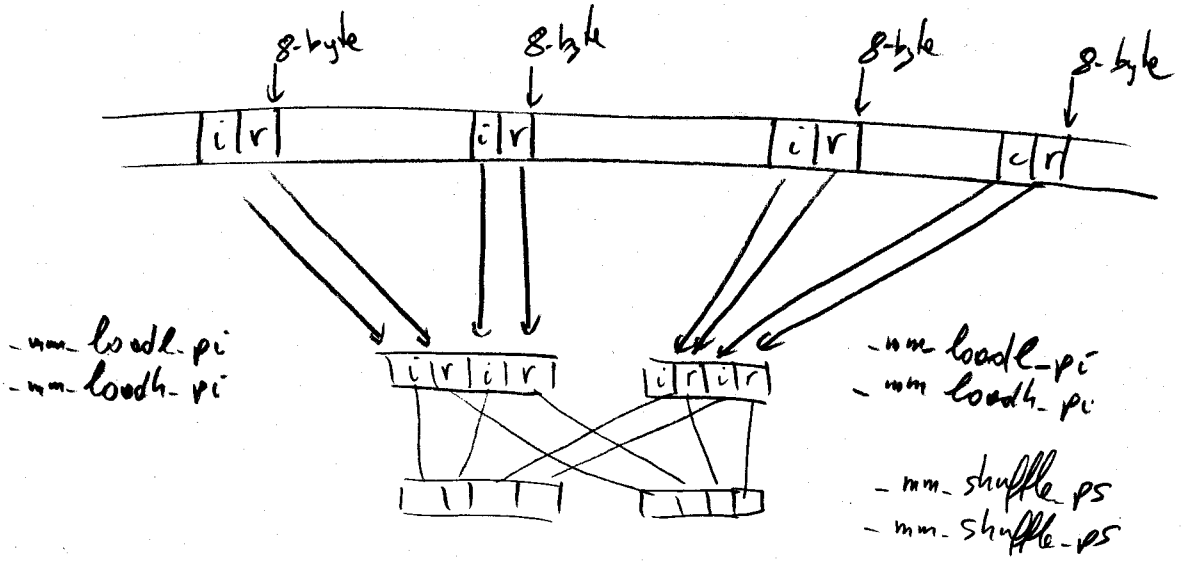
Example: Store 4 real Numbers



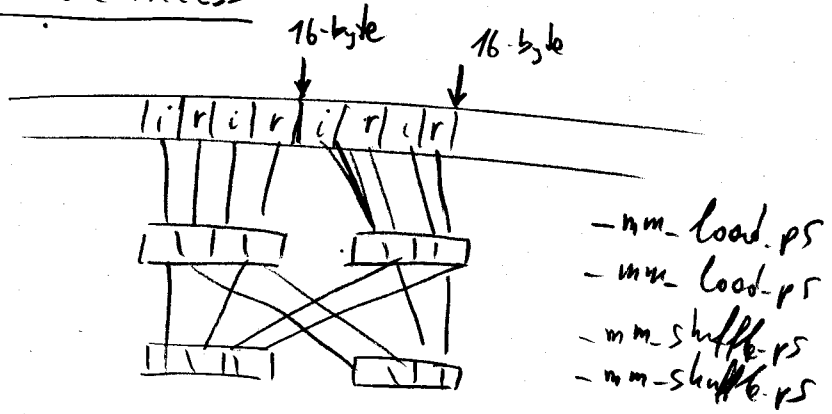

```
_mm_store_ss(m0, a);
b=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,1));
_mm_store_ss(m1, b);
c=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,2));
_mm_store_ss(m2, c);
d=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,3));
_mm_store_ss(m3, d);
```

Example: load 4 complex numbers

Strided access



Unit Stride Access



Formal Vectorization: Turning $A \otimes I_\nu$ into Code

Example: $A = \begin{pmatrix} 1 & 0.3 \\ 1 & -0.5 \end{pmatrix}$

$\nu = 4$ (SSE, Intel C++ compiler)

Code for A (scalar)

```
void A(float *y, float *x) {
  y[0] = x[0] + 0.3 * x[1];
  y[1] = x[0] - 0.5 * x[1];
}
```

Vector Code for $A \otimes I_4$

```
void AxI4(__m128 *y, __m128 *x) {
  y[0] = _mm_add_ps(x[0], _mm_mul_ps(_mm_set1_ps(0.3), x[1]));
  y[1] = _mm_sub_ps(x[0], _mm_mul_ps(_mm_set1_ps(0.5), x[1]));
}
```

Formal Vectorization of the WHT

$v = 2^{v'}$ vector length

$$WHT_{2^n} = \underbrace{WHT_2 \otimes \cdots \otimes WHT_2}_{n \text{ times}}$$

$$= WHT_{2^n/v} \otimes WHT_v$$

$$= \left(WHT_{2^n/v} \otimes \underline{I}_v\right)\left(\underline{I}_{2^n/v} \otimes WHT_v\right)$$

$$= \left(WHT_{2^n/v} \otimes \underline{I}_v\right)\left(\underline{I}_{2^n/v^2} \otimes \underline{I}_v \otimes WHT_v\right)$$

$$= \left(WHT_{2^n/v} \otimes \underline{I}_v\right)\left(\underline{I}_{2^n/v^2} \otimes L^{v^2}_v \left(WHT_v \otimes \underline{I}_v\right) L^{v^2}_v\right)$$

Tensor Product Rules

$$\left(A_m \otimes B_n\right) = \left(A_m \otimes \underline{I}_n\right)\left(\underline{I}_m \otimes A_n\right)$$

$$\left(A_m \otimes B_n\right) = L^{mn}_m \left(B_n \otimes A_m\right) L^{mn}_n$$

$$\underline{I}_{mn} = \underline{I}_m \otimes \underline{I}_n$$

The Bar Operator: Vectors and Matrices

complex Kronecker products do not capture memory access good enough

$\overline{|\cdot|}$ Operator

$$y = c \cdot x, \qquad x, y, c \in \mathbb{C}, \qquad c = a + ib$$
$$x = x_r + i x_i$$
$$y = y_r + i y_i$$

$$\downarrow$$

$$\bar{y} = \bar{c} \cdot \bar{x}, \qquad \bar{c} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

$$\bar{x} = \begin{pmatrix} x_r \\ x_i \end{pmatrix}, \qquad \bar{y} = \begin{pmatrix} y_r \\ y_i \end{pmatrix}$$

$$\overline{\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}} = \begin{pmatrix} \bar{x}_0 \\ \bar{x}_1 \\ \vdots \\ \bar{x}_{n-1} \end{pmatrix} = \begin{pmatrix} x_{0,r} \\ x_{0,i} \\ \vdots \\ x_{n-1,r} \\ x_{n-1,i} \end{pmatrix}$$

Interleaved complex format

$$\overline{\begin{pmatrix} a_{00} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \cdots & a_{m-1,n-1} \end{pmatrix}} = \begin{pmatrix} \bar{a}_{00} & \cdots & \bar{a}_{0,n-1} \\ \vdots & \ddots & \\ \bar{a}_{m-1,0} & \cdots & \bar{a}_{m-1,n-1} \end{pmatrix}$$

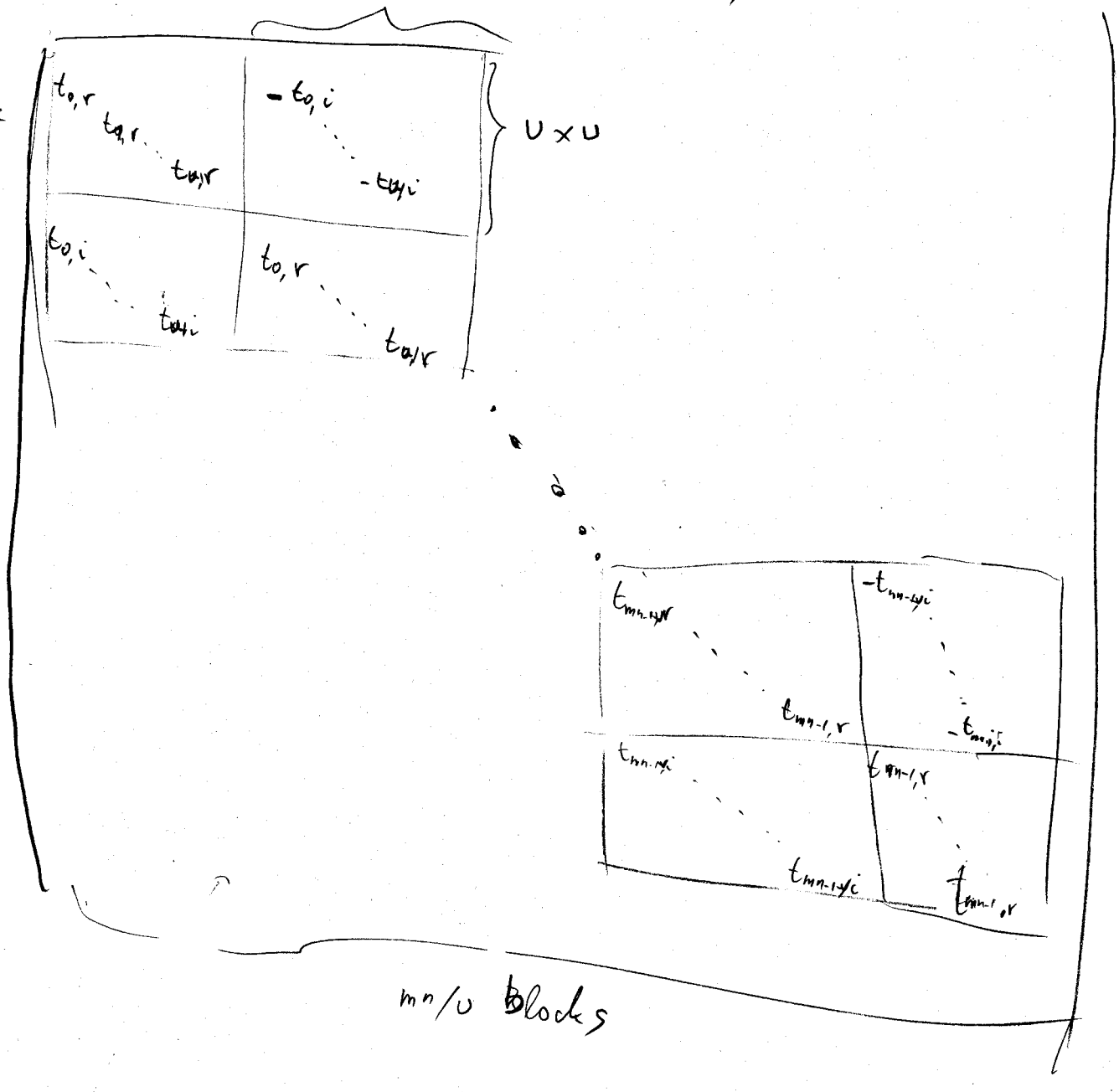
$$= \begin{pmatrix} \begin{vmatrix} a_{00,r} & -a_{00,i} \\ a_{00,i} & a_{00,r} \end{vmatrix} & \cdots & \begin{vmatrix} a_{0,n-1,r} & -a_{0,n-1,i} \\ a_{0,n-1,i} & a_{0,n-1,r} \end{vmatrix} \\ \vdots & \ddots & \\ \begin{vmatrix} a_{m-1,0,r} & -a_{m-1,0,i} \\ a_{m-1,0,i} & a_{m-1,0,r} \end{vmatrix} & \cdots & \begin{vmatrix} a_{m-1,n-1,r} & -a_{m-1,n-1,i} \\ a_{m-1,n-1,i} & a_{m-1,n-1,r} \end{vmatrix} \end{pmatrix}$$

The Bar Operator: Complex Diagonals

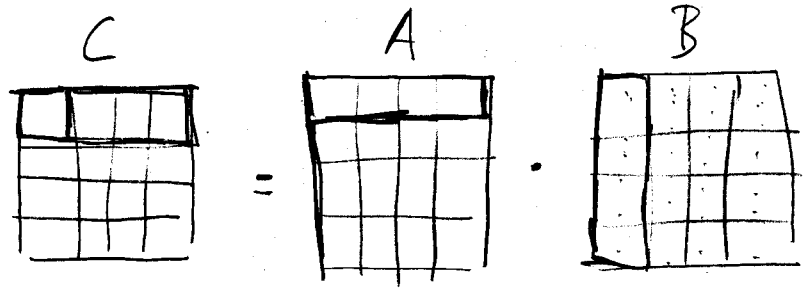
$$\overline{T}_n^{mn} = \left(I_{mn/v} \otimes L_v^{2v}\right) \widehat{T}_n^{mn} \left(I_{mn/v} \otimes L_2^{2v}\right)$$

$$\widehat{T}_n^{mn} = \left(I_{mn/v} \otimes L_2^{2v}\right) \overline{T}_n^{mn} \left(I_{mn/v} \otimes L_v^{2v}\right)$$

$=$

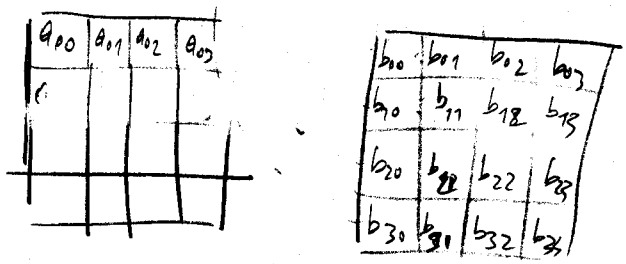


Example Matrix Multiplication: Vector loads

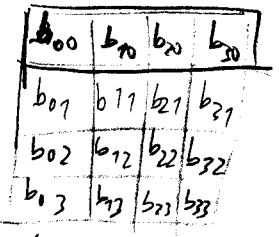



```
void mul4x4(__m128 *c, __m128 *a, __m128 *b) {
  __m128 b0, b1, b2, b3, c0, c1, c2, c3, a0, a1, a2, a3;
  ...

  b0 = b[0]; b1 = b[1]; b2 = b[2]; b3 = b[3];
  _MM_TRANSPOSE4_PS(b0, b1, b2, b3);
  a0 = a[0];
  c0 = _mm_mul_ps(b0, a0);
  c1 = _mm_mul_ps(b1, a0);
  c2 = _mm_mul_ps(b2, a0);
  c3 = _mm_mul_ps(b3, a0);
  _MM_TRANSPOSE4_PS(c0, c1, c2, c3);
  c[0] = _mm_add_ps(_mm_add_ps(c0, c1), _mm_add_ps(c2, c3));
  a1 = a[1];
  c0 = _mm_mul_ps(b0, a1);
  ...
  c3 = _mm_mul_ps(b3, a1);
  _MM_TRANSPOSE4_PS(c0, c1, c2, c3);
  c[1] = _mm_add_ps(_mm_add_ps(c0, c1), _mm_add_ps(c2, c3));
  ...

  c[3] = _mm_add_ps(...);
}
```

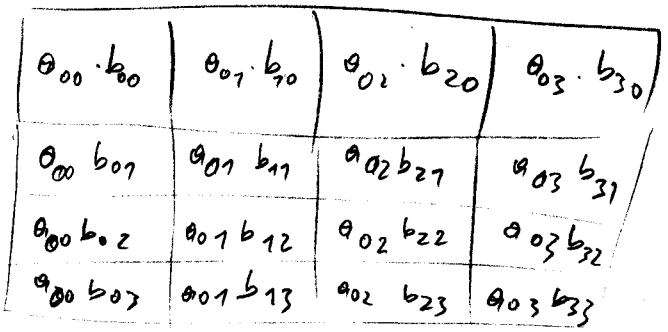
Example Matrix Multiplication Vector loads



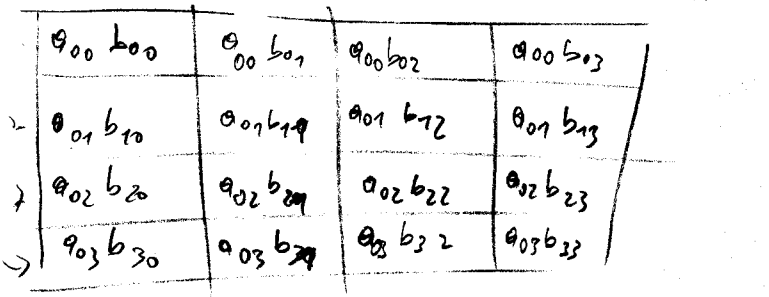
transpose



$*$



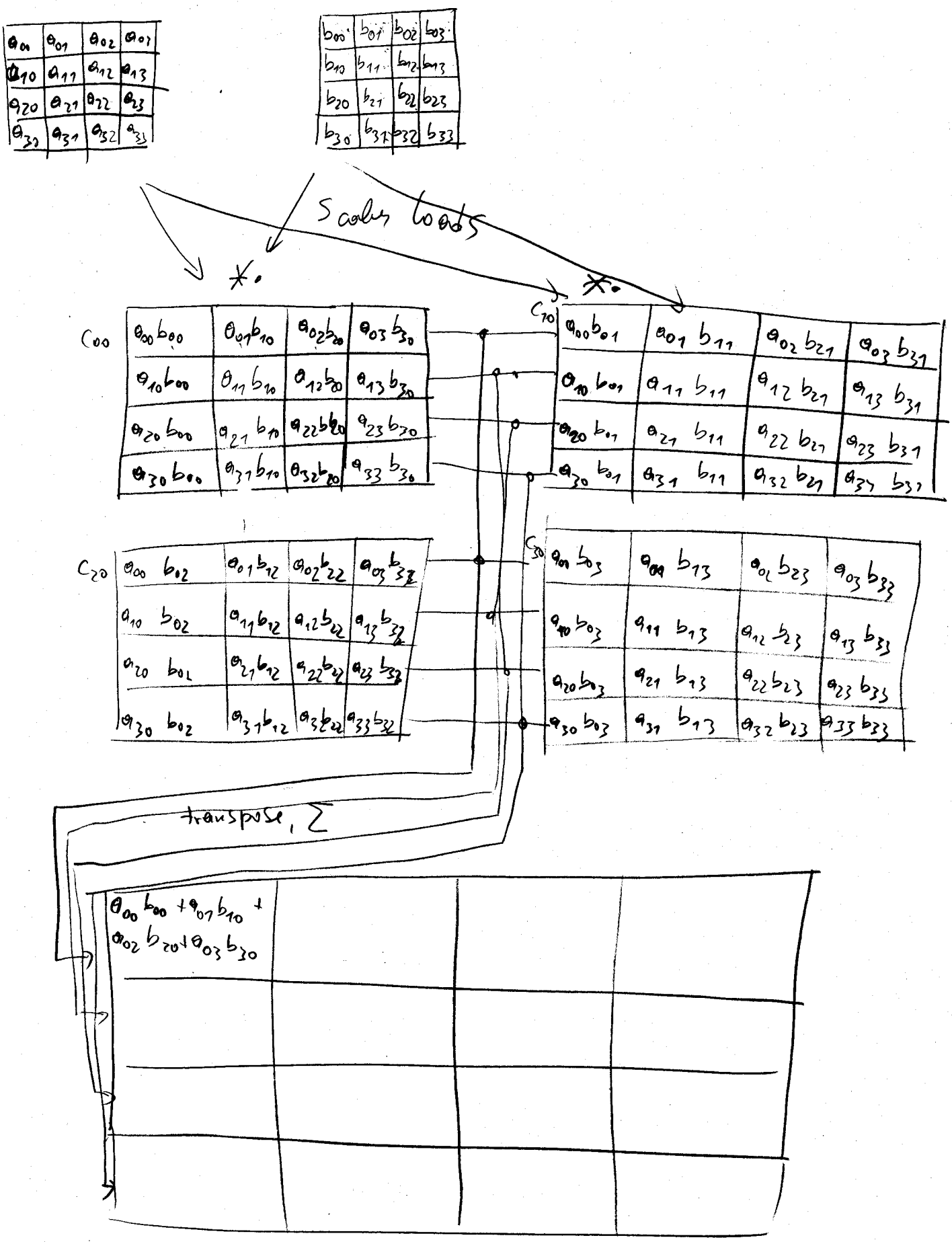
transpose



Σ

$a_{00} b_{00} + a_{01} b_{10} + a_{02} b_{20} + a_{03} b_{30}$	$a_{00} b_{01} + a_{01} b_{11} + a_{02} b_{21} + a_{03} b_{31}$	$a_{00} b_{02} + a_{01} b_{12} + a_{02} b_{22} + a_{03} b_{32}$	$a_{00} b_{03} + a_{01} b_{13} + a_{02} b_{23} + a_{03} b_{33}$

Example Matrix Multiplication Scalar Loads



Example Matrix Multiplication: Scalar loads

```
void mul_4x4(__m128 *C, __m128 a, float *b) {
  __m128 b0, b1, b2, b3, C00, ..., C33, a0, ..., a3;
  SCALAR_LOAD(b0, b, b+4, b+8, b+12);
  a0 = a[0]; a1 = a[1]; a2 = a[2]; a3 = a[3];
  C00 = _mm_mul_ps(a0, b0);
  C01 = _mm_mul_ps(a1, b0);
  C02 = _mm_mul_ps(a2, b0);
  C03 = _mm_mul_ps(a3, b0);
  SCALAR_LOAD(b1, b+1, b+5, b+9, b+13);
  C10 = _mm_mul_ps(a0, b1);
  ⋮
  C13 = _mm_mul_ps(a3, b1);
  SCALAR_LOAD(b2, b+2, b+6, b+10, b+14);
  ⋮
  SCALAR_LOAD(b3, b+3, b+7, b+11, b+15);
  _MM_TRANSPOSE4_PS(C00, C10, C20, C30);
  C[0] = _mm_add_ps(_mm_add_ps(C00, C10), _mm_add_ps(C20, C30));
  _MM_TRANSPOSE4_PS(C01, C11, C21, C31);
  C[1] = ...
  ⋮
}
```

Matrix Multiplication: Analysis

Vector Load

loads: 8
shuffles: 5 transposes = 40 instr
adds: 12
mults: 16
stores: 4

---

12 memops
40 shuffles
28 flops

Scalar Load

loads: 4 vec + 16 scalars
shuffles: 4 scalar loads = 12
4 transposes = 32
adds: 12
mults: 16
stores: 4

---

24 memops
44 shuffles
28 stores

Formal Vectorization: Vector Tensor Product

$Y = (A \otimes I_\nu) x$ can be vectorized easily:

Example: $A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$

$\nu = 2$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \left(\begin{array}{cc|cc} a_{00} & & a_{01} & \\ & a_{00} & & a_{01} \\ \hline a_{10} & & a_{11} & \\ & a_{10} & & a_{11} \end{array}\right) \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= \begin{pmatrix} a_{00} x_0 + a_{01} x_2 \\ a_{00} x_1 + a_{01} x_3 \\ a_{10} x_0 + a_{11} x_2 \\ a_{10} x_1 + a_{11} x_3 \end{pmatrix} = \begin{pmatrix} a_{00} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + a_{01} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \\ a_{10} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + a_{11} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} \end{pmatrix}$$

$$\begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_{00} \\ a_{00} \end{pmatrix} \vec{*} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \vec{+} \begin{pmatrix} a_{01} \\ a_{01} \end{pmatrix} \vec{*} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

$$\begin{pmatrix} y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} a_{10} \\ a_{10} \end{pmatrix} \vec{*} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \vec{+} \begin{pmatrix} a_{01} \\ a_{01} \end{pmatrix} \vec{*} \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$$

Constant vector

Contiguous in memory

$$\begin{pmatrix} a \\ b \end{pmatrix} \vec{+} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a+c \\ b+d \end{pmatrix}$$

$$\begin{pmatrix} a \\ b \end{pmatrix} \vec{*} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ bd \end{pmatrix}$$

Formal Vectorization: Parallel Tensor Product

$Y = (I_v \otimes A)\, x$ — <u>cannot</u> be vectorized easily

Example: $A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$

$v = 2$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \left(\begin{array}{cc|cc} a_{00} & a_{01} & & \\ a_{10} & a_{11} & & \\ \hline & & a_{00} & a_{01} \\ & & a_{10} & a_{11} \end{array}\right) \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= \begin{pmatrix} a_{00}x_0 + a_{01}x_1 \\ a_{10}x_0 + a_{11}x_1 \\ a_{00}x_2 + a_{01}x_1 \\ a_{10}x_2 + a_{11}x_3 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \end{pmatrix} x_0 + \begin{pmatrix} a_{01} \\ a_{11} \end{pmatrix} x_1 \\ \begin{pmatrix} a_{00} \\ a_{10} \end{pmatrix} x_2 + \begin{pmatrix} a_{01} \\ a_{11} \end{pmatrix} x_3 \end{pmatrix}$$

$$\begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_{00} \\ a_{10} \end{pmatrix} \vec{\times} \begin{pmatrix} x_0 \\ x_0 \end{pmatrix} \vec{+} \begin{pmatrix} a_{01} \\ a_{11} \end{pmatrix} \vec{\times} \begin{pmatrix} x_1 \\ x_1 \end{pmatrix}$$

$$\begin{pmatrix} y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} a_{00} \\ a_{01} \end{pmatrix} \vec{\times} \begin{pmatrix} x_2 \\ x_2 \end{pmatrix} \vec{+} \begin{pmatrix} a_{01} \\ a_{11} \end{pmatrix} \vec{\times} \begin{pmatrix} x_3 \\ x_3 \end{pmatrix}$$

$\neq \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}$ ↯

↑ Vector constants ✓

Formal Vectorization: Parallel Tensor Product

$$\left(I_m \otimes A_n\right) = \left(I_{m/v} \otimes I_v \otimes A_n\right)$$

$$= \left(I_{m/v} \otimes L^{vn}_{v} \left(A_n \otimes I_v\right) L^{vn}_{n}\right)$$

$$= \left(I_{m/v} \otimes \left(L^{n}_{v} \otimes I_v\right)\left(I_{n/v} \otimes L^{v^2}_{v}\right)\left(A_n \otimes I_v\right)\left(I_{n/v} \otimes L^{v^2}_{v}\right)\left(L^{n}_{n/v} \otimes I_v\right)\right)$$

with

$$\left(A_m \otimes B_n\right) = L^{mn}_{m} \left(B_n \otimes A_m\right) L^{mn}_{n}$$

$$L^{kmn}_{n} = \left(L^{kn}_{n} \otimes I_m\right)\left(I_k \otimes L^{mn}_{n}\right)$$

$$L^{kmn}_{km} = \left(I_k \otimes L^{mn}_{m}\right)\left(L^{kn}_{k} \otimes I_m\right)$$

$$L^{nv}_{n} = \left(I_{n/v} \otimes L^{v^2}_{v}\right)\left(L^{n}_{n/v} \otimes I_v\right)$$

$$L^{nv}_{v} = \left(L^{n}_{v} \otimes I_v\right)\left(I_{n/v} \otimes L^{v^2}_{v}\right)$$

The Short Vector Cooley-Tukey FFT

$$\overline{DFT_{mn}} = \left(I_{mn/\nu} \otimes L^{2\nu}_{\nu}\right)\left(\overline{DFT_m} \otimes I_{n/\nu} \otimes I_\nu\right) \hat{T}^{mn}_n$$
$$\left(I_{m/\nu} \otimes \left(L^{2n}_{\nu} \otimes I_\nu\right)\left(I_{2n/\nu} \otimes L^{\nu^2}_{\nu}\right)\left(I_{n/\nu} \otimes L^{2\nu}_{2} \otimes I_\nu\right)\left(\overline{DFT_n} \otimes I_\nu\right)\right)$$
$$\left(I_{mn/\nu} \otimes L^{2\nu}_{2}\right)\left(L^{mn/\nu}_{m/\nu} \otimes I_2 \otimes I_\nu\right)$$

Objects in formula:

$A \otimes I_\nu \Rightarrow$

$t_j = t_k \rightarrow \vec{t}_j = \vec{t}_k$
$a + b \rightarrow \vec{a} + \vec{b}$
$a * b \rightarrow \vec{a} * \vec{b}$

$I_k \otimes L^{\nu^2}_{\nu} \Rightarrow$ _MM_TRANSPOSE4_PS( )
⋮
_MM_TRANSPOSE4_PS( )

$I_k \otimes L^{2\nu}_{2} \Rightarrow$ $a =$ _mm_shuffle_ps( ) Interleaved complex
$b =$ _mm_shuffle_ps( ) → split complex

$I_k \otimes L^{2\nu}_{\nu} \Rightarrow$ $a =$ _mm_unpacklo_ps( ) split complex
$b =$ _mm_unpacklo_ps( ) → interleaved complex

$\hat{T}^{mn}_n \Rightarrow$ $\vec{y}_0 = \begin{pmatrix} t_{0,r} \\ \vdots \\ t_{\nu-1,r} \end{pmatrix} \vec{*} \vec{x}_0$
$\vec{y}_3 =$ ⋮
$\vec{y}_4 = \vec{y}_0 \vec{+} \vec{y}_1$
⋮

4 mul, 2 add
or
3 mul, 3 add
per $2\nu^2 \times 2\nu^2$ block