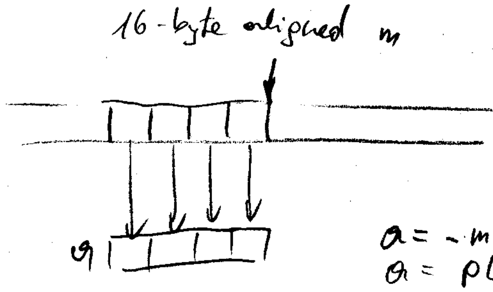
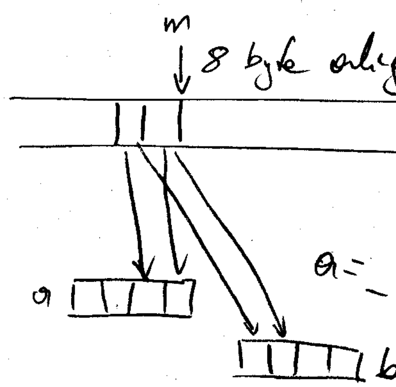
Table of Contents

Load Instructions

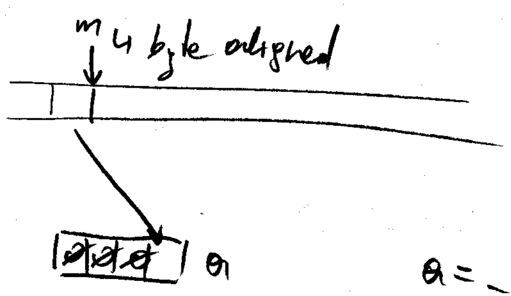


a = _mm_load_ps(m); aligned, explicit
a = p[i]; — " —, implicit
a = _mm_loadu_ps(m); unaligned



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);

4.0	3.0	2.0	1.0

d = _mm_set1_ps(1.0);

1.0	1.0	1.0	1.0

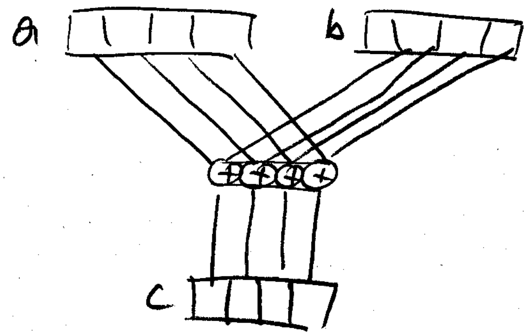
e = _mm_set_ss(1.0);

0	0	0	1.0

f = _mm_setzero_ps();

0	0	0	0

Vector arithmetic

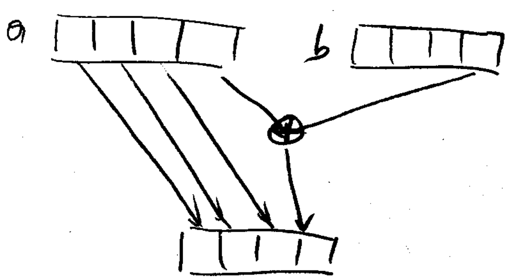


c = _mm_add_ps(a, b)

Same:

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

Scalar arithmetic

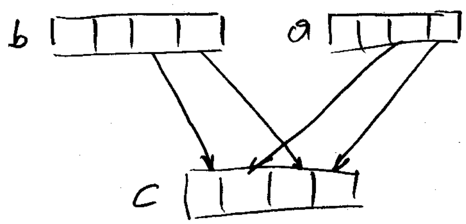


c = _mm_add_ss(a, b)

Reorder Instructions

Unpack lo

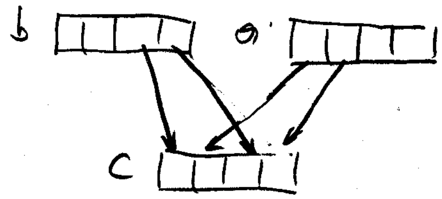
$c = \_mm\_unpacklo\_ps(a,b)$



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

Unpack hi

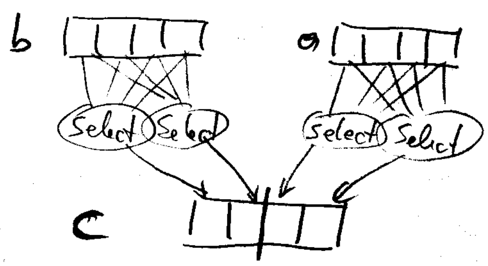
$c = \_mm\_unpackhi\_ps(a,b)$



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

Shuffle

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

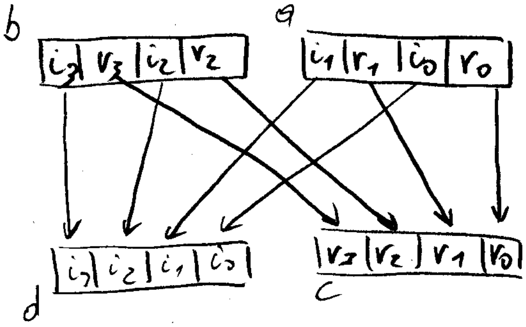


$c_0 = a_i$
$c_1 = a_j$
$c_2 = b_k$
$c_3 = b_l$

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

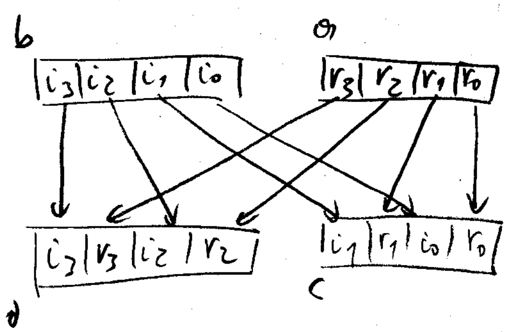
Reorder Instructions – Examples

Interleaved Complex → split complex : L^8_2



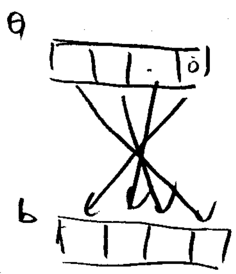

```
c = _mm_shuffle_ps(a, b, _MM_SHUFFLE(2,0,2,0));
d = _mm_shuffle_ps(a, b, _MM_SHUFFLE(3,1,3,1));
```

Split complex → interleaved complex : L^8_4




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

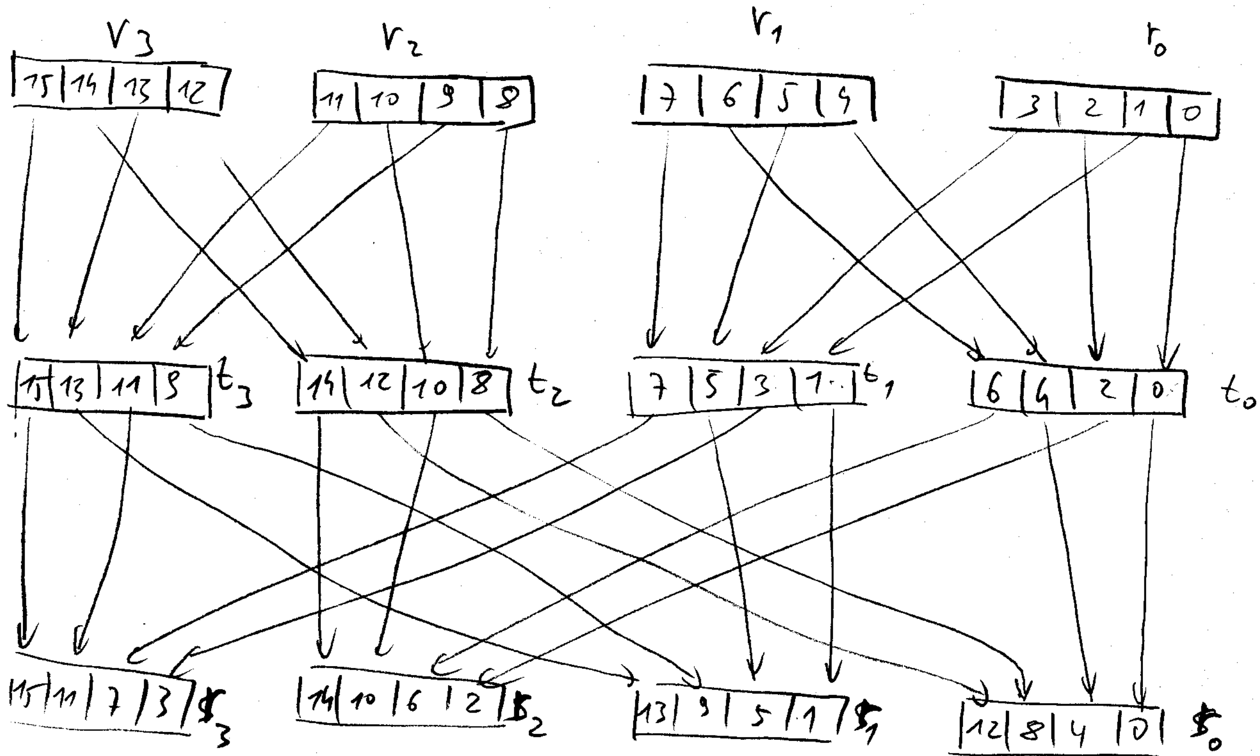
Reverse Vector : J_4




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

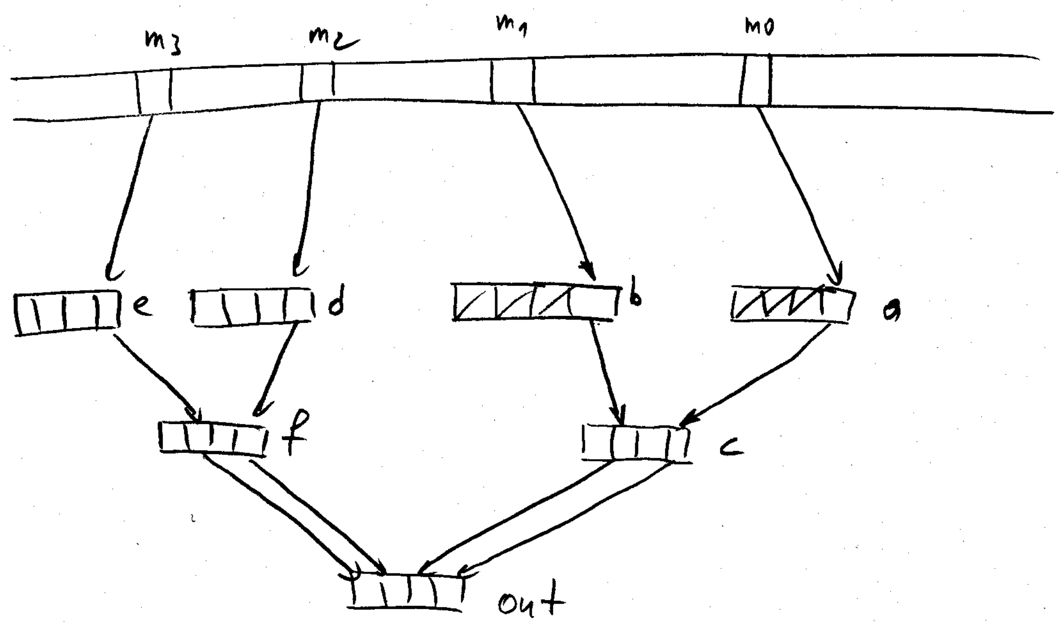
Reorder Instructions - Examples

Transpose 4x4 : $L^{16}_{4} = \left(L^{4}_{2} \otimes I_4\right)\left(I_2 \otimes L^{8}_{2}\right)\left(L^{4}_{2} \otimes I_4\right)\left(I_2 \otimes L^{8}_{2}\right)$



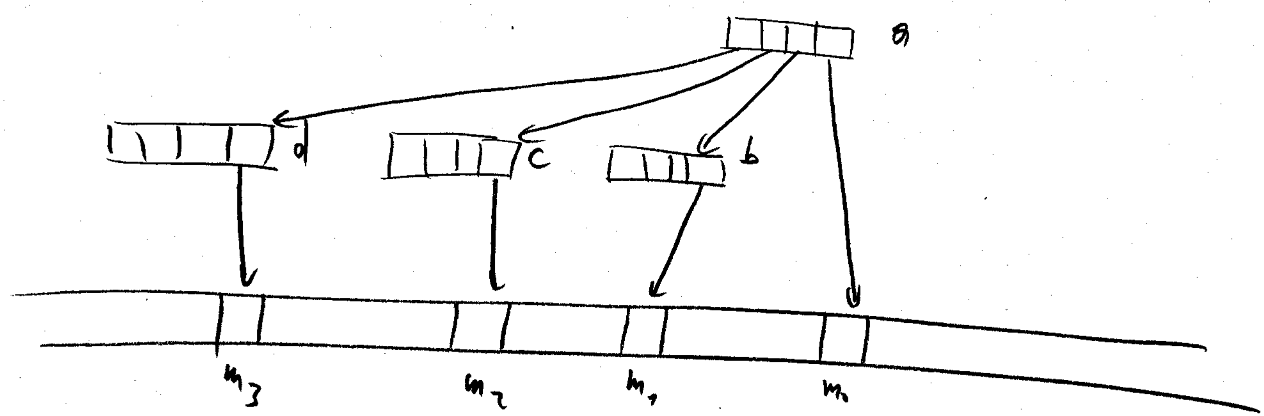

```
#define _MM_TRANSPOSE4_PS(r0,r1,r2,r3) {                  \
  __m128 t0,t1,t2,t3;                                     \
  t0 = _mm_shuffle_ps(r0, r1, _MM_SHUFFLE(2,0,2,0));      \
  t1 = _mm_shuffle_ps(r0, r1, _MM_SHUFFLE(3,1,3,1));      \
  t2 = _mm_shuffle_ps(r2, r3, _MM_SHUFFLE(2,0,2,0));    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

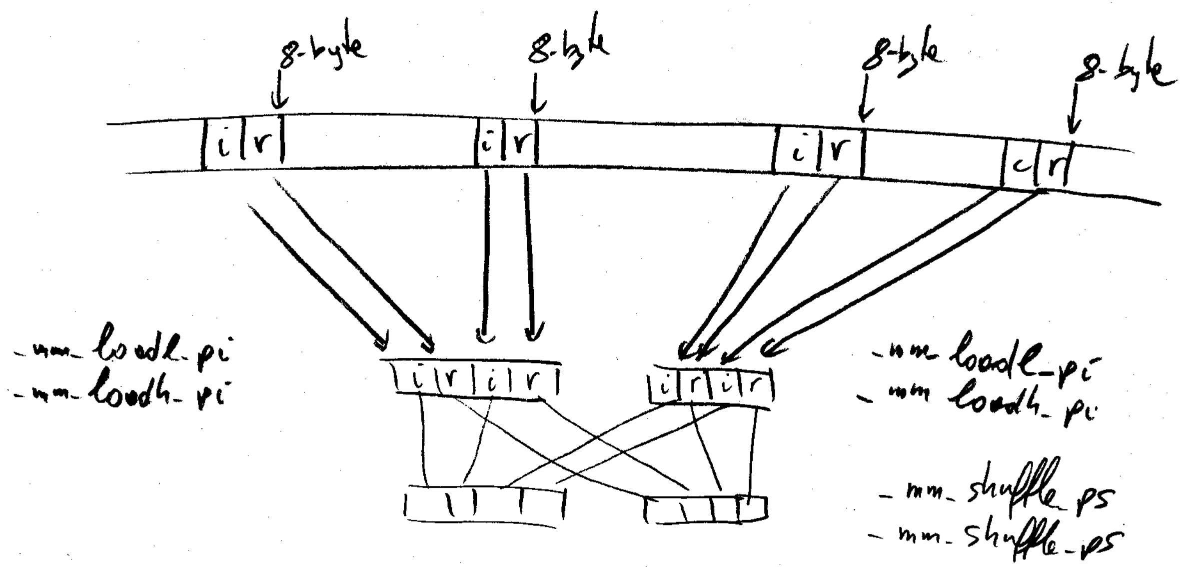
Example: Store 4 real Numbers



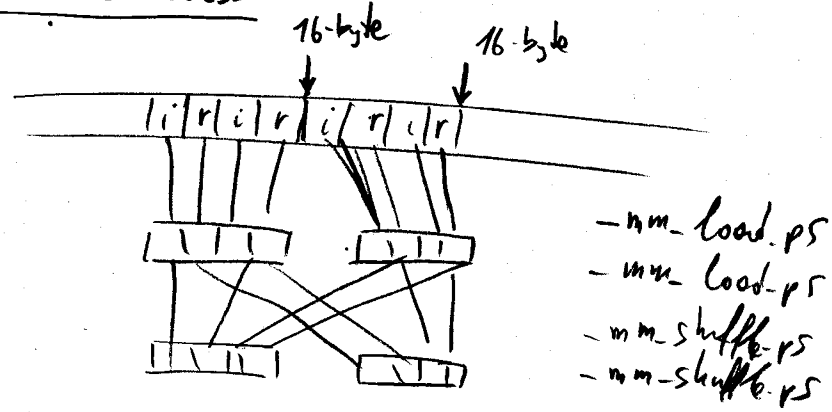

```
_mm_store_ss(m0, a);
b=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,1));
_mm_store_ss(m1, b);
c=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,2));
_mm_store_ss(m2, c);
d=_mm_shuffle_ps(a, a, _MM_SHUFFLE(0,0,0,3));
_mm_store_ss(m3, d);
```

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

$\nu = 2^{\nu'}$ vector length

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

$$= WHT_{2^n/\nu} \otimes WHT_\nu$$

$$= \left(WHT_{2^n/\nu} \otimes I_\nu\right)\left(I_{2^n/\nu} \otimes WHT_\nu\right)$$

$$= \left(WHT_{2^n/\nu} \otimes I_\nu\right)\left(I_{2^n/\nu^2} \otimes I_\nu \otimes WHT_\nu\right)$$

$$= \left(WHT_{2^n/\nu} \otimes I_\nu\right)\left(I_{2^n/\nu^2} \otimes L^{\nu^2}_\nu \left(WHT_\nu \otimes I_\nu\right) L^{\nu^2}_\nu\right)$$

Tensor Product Rules

$$(A_m \otimes B_n) = (A_m \otimes I_n)(I_m \otimes A_n)$$

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

$$I_{mn} = I_m \otimes I_n$$

The Bar Operator: Vectors and Matrices

complex Kronecker products do not capture memory access good enough

$\overline{\cdot}$ Operator

$y = c \cdot x$, $\quad x, y, c \in \mathbb{C}$, $\quad c = a + ib$

$x = x_r + i x_i$

$y = y_r + i y_i$

$\downarrow$

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

$\overline{x} = \begin{pmatrix} x_r \\ x_i \end{pmatrix}$, $\quad \overline{y} = \begin{pmatrix} y_r \\ y_i \end{pmatrix}$

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

Interleaved complex format

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

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

The Box Operator: Complex Diagonals

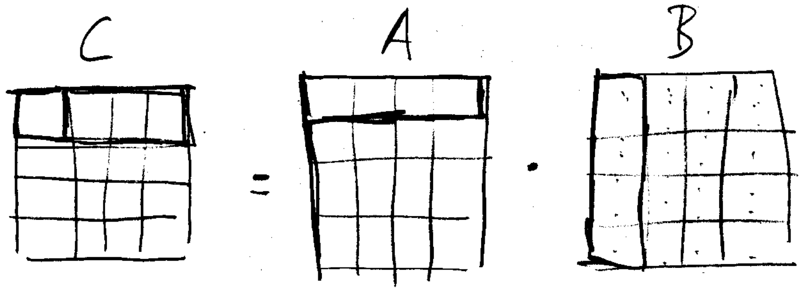
$$\overline{T}_n^{mn} = \left(I_{mn/v} \otimes L_v^{2v}\right) \widehat{T}_n^{mn} \left(I_{mn/v} \otimes L_2^{2v}\right)$$

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

$$= \underbrace{\begin{bmatrix} \begin{array}{cc|cc} \begin{matrix} t_{0,r} & & \\ & t_{1,r} & \\ & & \ddots \\ & & & t_{v-1,r} \end{matrix} & & \begin{matrix} -t_{0,i} & & \\ & \ddots & \\ & & -t_{v-1,i} \end{matrix} & \\ \hline \begin{matrix} t_{0,i} & & \\ & \ddots & \\ & & t_{v-1,i} \end{matrix} & & \begin{matrix} t_{0,r} & & \\ & \ddots & \\ & & t_{v-1,r} \end{matrix} & \end{array} & & \\ & \ddots & \\ & & \begin{array}{c|c} \begin{matrix} t_{mn-v,r} & & \\ & \ddots & \\ & & t_{mn-1,r} \end{matrix} & \begin{matrix} -t_{mn-v,i} & & \\ & \ddots & \\ & & -t_{mn-1,i} \end{matrix} \\ \hline \begin{matrix} t_{mn-v,i} & & \\ & \ddots & \\ & & t_{mn-1,i} \end{matrix} & \begin{matrix} t_{mn-1,r} & & \\ & \ddots & \\ & & t_{mn-1,r} \end{matrix} \end{array} \end{bmatrix}}_{mn/v \text{ Blocks}}$$

$v \times v$

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(a0, a1, a2, a3);
  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

a_{00}	a_{01}	a_{02}	a_{03}

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

transpose

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

$*$

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

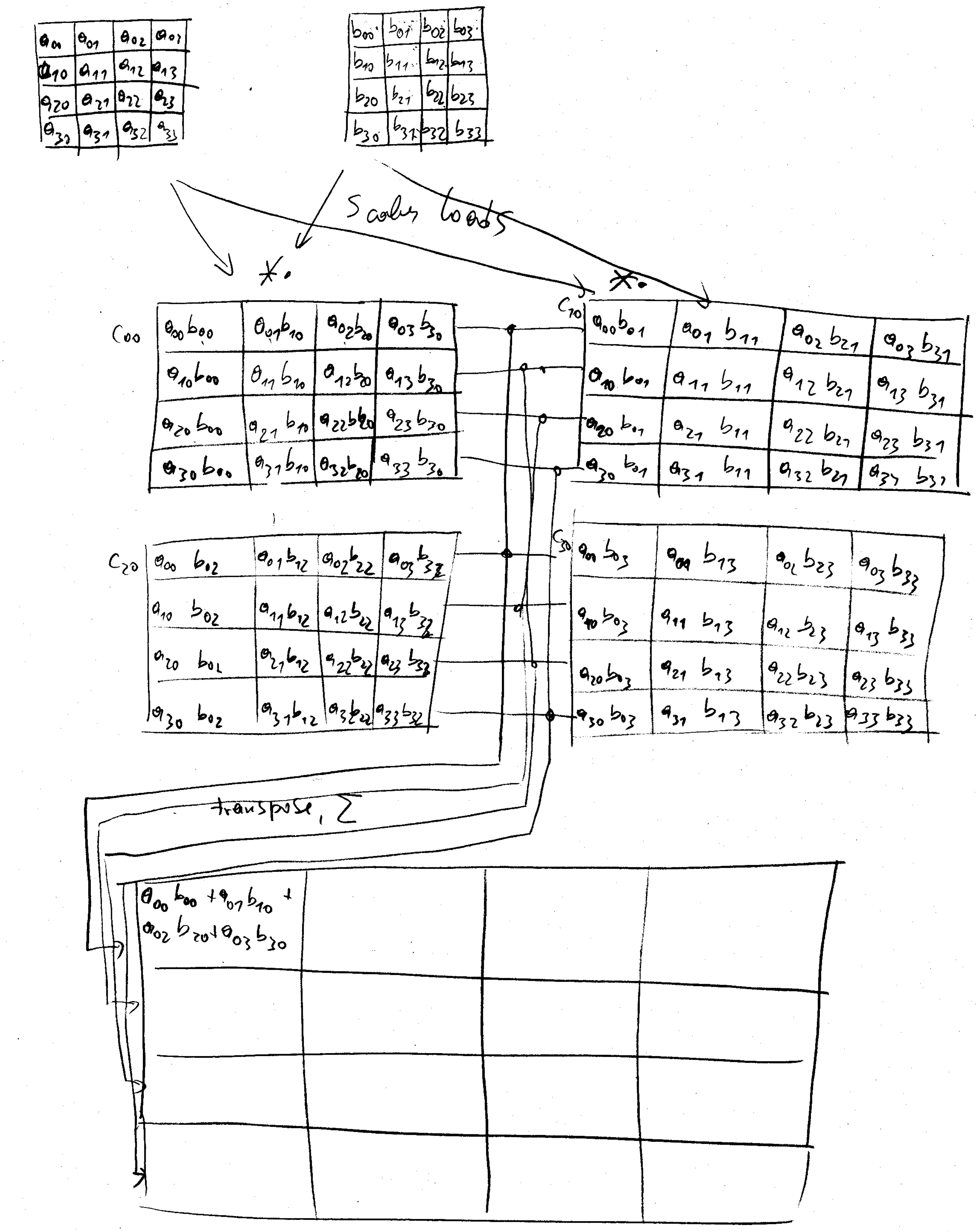
transpose

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

$\sum$

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

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 mem ops
40 shuffles
28 flops

Scalar Load

loads : 4 vect 16 scalar
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: $A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{10} \end{pmatrix}$

$\nu = 2$

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

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

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

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

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

$y = (I_v \otimes A)\, x$ **cannot** be vectorized easily

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

$v = 2$

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

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

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

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

$\neq \begin{pmatrix} x_0 \\ y_1 \end{pmatrix}, \begin{pmatrix} y_2 \\ x_3 \end{pmatrix}$ ⚡

Vector constants ✓

Formal Vectorization: Parallel Tensor Product

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

with

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

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

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

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

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

The Short Vector Cooley-Tukey FFT

$$\overline{DFT}_{mn} = \left(I_{mn/v} \otimes L^{2v}_{v}\right)\left(\overline{DFT_{m} \otimes I_{n/v}} \otimes I_{v}\right)\hat{T}^{mn}_{n}$$

$$\left(I_{m/v} \otimes \left(L^{2n}_{v} \otimes I_{v}\right)\left(I_{2n/v} \otimes L^{v^2}_{v}\right)\left(I_{n/v} \otimes L^{2v}_{2} \otimes I_{v}\right)\left(\overline{DFT}_{n} \otimes I_{v}\right)\right)$$

$$\left(I_{mn/v} \otimes L^{2v}_{2}\right)\left(L^{mn/v}_{m/v} \otimes I_{2} \otimes I_{v}\right)$$

Objects in formulas:

$A \otimes I_v \Rightarrow$

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

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

⋮

_MM_TRANSPOSE4_PS( )

$I_k \otimes L^{2v}_{2} \Rightarrow$ a = _mm_shuffle_ps( )
b = _mm_shuffle_ps()

Interleaved complex → split complex

$I_k \otimes L^{2v}_{v} \Rightarrow$ a = _mm_unpacklo_ps()
b = _mm_unpacklo_ps()

split complex → interleaved complex

$\hat{T}^{mn}_{n} \Rightarrow$

$$\vec{y}_0 = \begin{pmatrix} t_{0,r} \\ \vdots \\ t_{v-1,r} \end{pmatrix} \vec{*}\, \vec{x}_0$$

$$\vdots$$

$$\vec{y}_3 = \vdots$$

$$\vec{y}_4 = \vec{y}_0 \,\vec{+}\, \vec{y}_1$$

$$\vdots$$

4 mul, 2add or

3 mul, 3add

per $2v^2 \times 2v^2$ block