Assignment 3 – Task 1: Dense MVM

- All data in L1

  This is a 2 GHz core, 1 LD/cy, 1 MULT/cy, 1 ADD/cy

  → P_{max} = 2 Flops / 2 cy = 1 Flop/cy = 2 Gflop/s

  In-core bottleneck: LD throughput

- Without unrolling?

  → This is a scalar product, ADD latency = 4 cy
  → P = 2 Flops / 4 cy = 1 Gflop/s

  In-core bottleneck: ADD latency due to pipeline stall

Assignment 3 – Task 1: Dense MVM

- What would change with 4-way SIMD?
  - Locality analysis unchanged
  - Max performance in L1: 4x boost (still LOAD bound)
  - Naïve code performance: unchanged (no SIMD without unrolling)

Assignment 3 – Task 1: Dense MVM

(scalar product style)

- Locality of access (**a[][]** in memory):

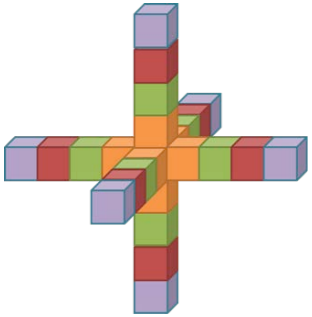
```
for(i=0; i<N; ++i)
  for(j=0; j<N; ++j)
    c[i] += a[i][j] * b[j];
```

a[][] purely spatial locality (each element accessed exactly once in linear order)

c[] spatial and temporal locality: all elements accessed in linear order, each element reused N-1 times

b[] spatial locality: all elements accessed in linear order
temporal locality: N-1 times reuse if array b fits into some cache

Assignment 3 – Task 2: Long-range stencil P_{max}



```
for(int k=4; k < N-4; k++) {
 for(int j=4; j < N-4; j++) {
  for(int i=4; i < N-4; i++) {
   U[k][j][i] = 2*V[k][j][i] - U[k][j][i] + C[k][j][i] *
   ( c0* V[k][j][i] +
     c1*(V[ k ][ j ][i+1]+V[ k ][ j ][i-1] +
         V[ k ][j+1][ i ]+V[ k ][j-1][ i ] +
         V[k+1][ j ][ i ]+V[k-1][ j ][ i ])+
     c2*(V[ k ][ j ][i+2]+V[ k ][ j ][i-2] +
         V[ k ][j+2][ i ]+V[ k ][j-2][ i ] +
         V[k+2][ j ][ i ]+V[k-2][ j ][ i ])+
     c3*(V[ k ][ j ][i+3]+V[ k ][ j ][i-3] +
         V[ k ][j+3][ i ]+V[ k ][j-3][ i ] +
         V[k+3][ j ][ i ]+V[k-3][ j ][ i ])+
     c4*(V[ k ][ j ][i+4]+V[ k ][ j ][i-4] +
         V[ k ][j+4][ i ]+V[ k ][j-4][ i ] +
         V[k+4][ j ][ i ]+V[k-4][ j ][ i ])
   );
  }
 }
}
```

LOAD:	25+1+1
STORE:	1
ADD/SUB:	26
MULT:	7

The hardware

Microarchitecture	SandyBridge-EP	IvyBridge-EP	Haswell-EP
Shorthand	SNB	IVB	HSW
Xeon Model	E5-2680	E5-2690 v2	E5-2695 v3
Year	03/2012	09/2013	09/2014
Clock speed (fixed)	2.7 GHz	2.2 GHz	2.3 GHz
Cores/Threads	8/16	10/20	14/28
Load/Store throughput per cycle			
AVX(2)	1 LD & 1/2 ST	1 LD & 1/2 ST	2 LD & 1 ST
SSE/scalar	2 LD ‖ 1 LD & 1 ST	2 LD ‖ 1 LD & 1 ST	2 LD & 1 ST
L1 port width	2×16+1×16 B	2×16+1×16 B	2×32+1×32 B
ADD throughput	1 / cy	1 / cy	1 / cy
MUL throughput	1 / cy	1 / cy	2 / cy
FMA throughput	n/a	n/a	2 / cy
L2-L1 data bus	32 B	32 B	64 B
L3-L2 data bus	32 B	32 B	32 B
LLC size	20 MiB	25 MiB	35 MiB
Main memory	4×DDR3-1600	4×DDR3-1866	4×DDR4-2133
Peak memory BW	51.2 GB/s	51.2 GB/s	68.3 GB/s
Load-only BW	43.6 GB/s (85%)	46.1 GB/s (90%)	60.6 GB/s (89%)
T_{L3Mem} per CL	3.96 cy	3.05 cy	2.43 cy

Assignment 3 – Task 2: Long-range stencil P_{max}

LOAD:	25+1+1
STORE:	1
ADD/SUB:	26
MULT:	7

Cycles per scalar iteration (33 Flops):

		Scalar [cy/it]	SSE [cy/it]	AVX [cy/it]
Ivy Bridge	LD/ST ADD MULT Overall bottleneck	13+1 26 7 **ADD (26)**	(13+1)/2 26/2 7/2 **ADD (13)**	27/4 26/4 7/4 **LD (6.75)**
Haswell (no FMA)	LD/ST ADD MULT Overall bottleneck	13.5 26 3.5 **ADD (26)**	13.5/2 26/2 3.5/2 **ADD (13)**	13.5/4 26/4 3.5/4 **ADD (6.5)**
Haswell (MULT/ADD as FMA)	LD/ST ADD or MULT Overall bottleneck	13.5 33/2 **FMA (16.5)**	13.5/2 33/4 **FMA (8.25)**	13.5/4 33/8 **FMA (4.125)**

Assignment 3 – Task 3: Bad code performance

- **Problems in this code**
  - C uses “row major storage” for matrices, so loop order & index order cause strided data access to **`a[][]`**
  - This stride is hazardous:
    - Every CL load transfers 64 byte, but only 8 byte are used
    - 16384 CLs loaded in one **`j`** loop traversal → 1 MiB of data
      - With 25 MiB of L3 cache this data can be picked up in the next **`i`** iteration, but it’s still far away from the core
    - Even worse: Power of 2 in leading dimension reduces effective cache size
      - *m*-way set-associative cache → effective size *m* CLs
      - Cache thrashing
    - SIMD vectorization will be inefficient
      - Successive **`j`** iterations load data from non-consecutive addresses
        → Even if execution in SIMD parallel, data transfer is not

```
#define N 16384
//...
s = 0.0;
for (int i=0; i<N; i++) {
  for(int j=0; j<N; j++) {
    s += a[j][i];
  }
}
```

Assignment 3 – Task 3: Bad code performance

- **Fixing it**
  - Interchange loops → optimal data access
  - Leading dimension problem is eliminated automatically (no strided access → no thrashing)

```
#define N 16384
//...
s = 0.0;
for (int j=0; i<N; i++) {
  for(int i=0; j<N; j++) {
    s += a[j][i];
  }
}
```

- **What to do if loop interchange cannot fix it?**

```
#define N 16384
//...
s = 0.0;
for (int j=0; i<N; i++) {
  for(int i=0; j<N; j++) {
    b[i][j] = a[j][i];
  }
}
```