Assignment 2 – Task 1

```
double s=0.0;
double a[N],b[N];
// ... initialization omitted
for(i=0; i<N; ++i)
  s = s + a[i] * b[i];
```

- Scalar product in L1 cache
- Core
  - Per cycle: 1 LD, 1 ADD, 1 MULT
  - ADD (MULT) latency: 3(5) cycles
  - L1 Load latency: 4 cycles
  - Horizontal AVX add (DP): 8 cycles
  - No other restrictions
- Overall pipeline latency (time to first result)
  - L = 4+1 cy (LD) + 5 cy (MULT) + 3 cy (ADD) = 13 cy
- Steady state, no unrolling, no SIMD
  - 1 iteration == 2LD, 1 ADD, 1 MULT → 3 cycles/Iteration → 2/3 F/cy
- N=8 performance?
  - $\frac{N}{T(N)} = \frac{N}{L+3(N-1)}$ → 8/34 Iteration/cy = 0.47 F/cy
- Half-max: $\frac{N}{L+3(N-1)} = \frac{1}{2} \times \frac{1}{3}$ → solve for N → $N = \frac{L}{3} - 1 \approx 3$

Assignment 2 – Task 1

- SIMD vectorization w/ AVX (4-way), no unrolling
  - Maximum (steady-state) throughput:

    1 AVX iteration == 2LD, 1 ADD, 1 MULT
    → 3 cycles/Iteration → 4·2/3 F/cy = 8/3 F/cy

  - Performance @ N=8: Now we have L=13+8=21, and N/4 AVX iterations:

    [N/4]/T([N/4]) = [N/4] / (L+3*[N/4-1])
    → 1/12 AVX Iteration/cy = 2/3 F/cy

    For half maximum throughput we need N = 4L/3-4 ≈ 24 iterations!

- SIMD vectorization w/ 2xAVX (8-way), no unrolling
  - Maximum throughput doubles
  - Half-max throughput: N = 8L/3-8 ≈ 48 iterations!

Assignment 2 – Task 1

- AVX vectorization (4-way) plus 2-way modulo unrolling
  - Max performance:

    Only one bubble in the ADD pipeline left
    → ADD limitation at 3 cycles per 2 (AVX) iterations

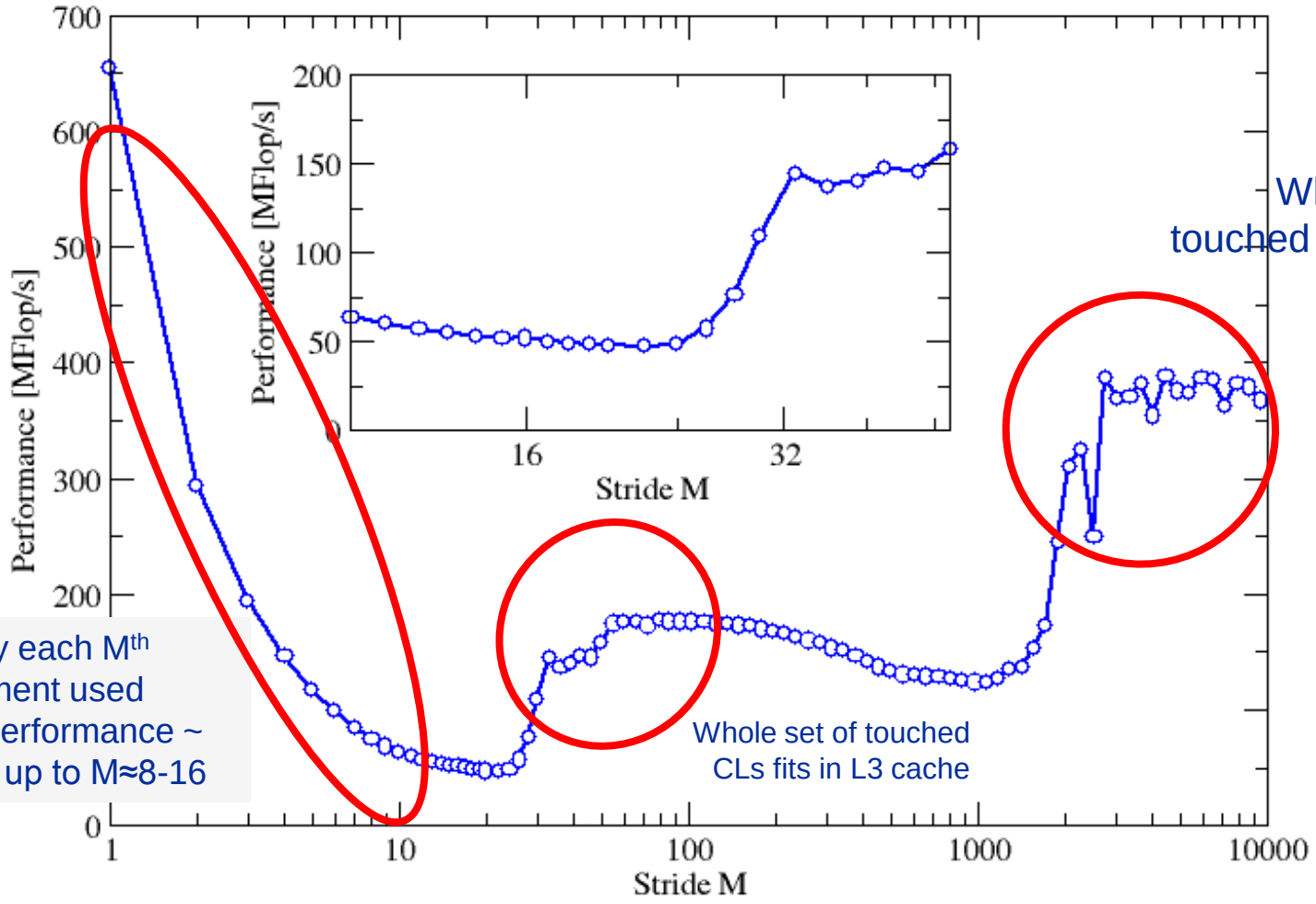
    BUT: 2 iterations need 4 Loads
    → LD limitation at 4 cycles per 2 (AVX) iterations

    → Overall limitation is the LD pipeline!
    → Max performance = 16 F / 4 cycles = 4 F/cycle

- 4-way unrolling would change nothing.

Assignment 2 – Task 2

- Strided vector triad ($N=2x10^6$) on one Emmy core



Only each Mth element used → performance ~ 1/M up to M≈8-16

Whole set of touched CLs fits in L3 cache

Whole set of touched CLs fits in L2 cache

Assignment 2 – Task 3

- **Problems in this code?**

```
double mat[8192][8192],s[8192][8192];
// ...
srand(1); // random seed
for(i=0; i<N ; ++i) {
  val = (double)rand(); // random number
  for(j=0; j<N; ++j) {
    mat[j][i] = s[j][i]/val;
  }
}
```

- Power of two in leading array dimension
- Wrong loop order □strided access to m and s
- Divide in inner loop

Consequence: non-vectorizable, bad BW utilization, bad pipelining

Assignment 2 – Task 3

- Optimizations
  - Move divide out of inner loop

```
srand(1); // random seed
for(i=0; i<N ; ++i) {
  val = 1./(double)rand();
  for(j=0; j<N; ++j) {
    mat[j][i] = s[j][i]* val;
  }
}
```

  - Tabulate random numbers, interchange loops, tabulation inside timing

```
timing();
  srand(1); // random seed
  for(i=0; i<N ; ++i)
    table[i] = 1./(double)rand();
  for(j=0; j<N ; ++j) {
    for(i=0; i<N; ++i) {
      mat[j][i] = s[j][i]*table[i];
    }
}
timing();
```

Assignment 2 – Task 3

Results for Intel Ivy Bridge core @ 2.2 GHz (Emmy)