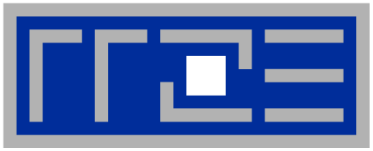
Assignment 2 – Task 1



- Compiling with **`-no-vec`**: 14 cy/it

  → Scalar divide instruction inv. throughput = 14 cy/instr.
  → AVX divide instruction inv. throughput = 28 cy/instr.

  Conclusion: Intel has saved hardware when designing the DIV units. There are really only two dividers even though the AVX divide instruction is 4-wide. An AVX divide is carried out internally as a sequence of two 16-byte wide divides.

Assignment 2 – Task 2

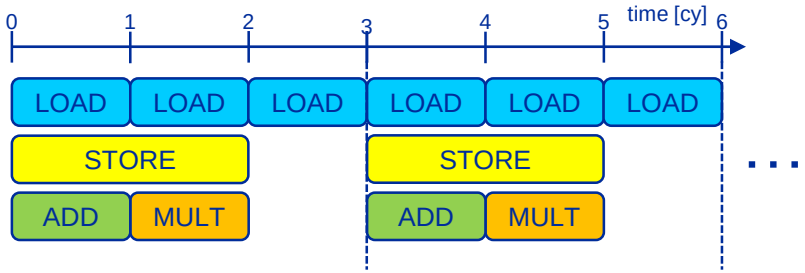
- $P_{\max}$ calculation for vector triad on Ivy Bridge

```
A(:) = B(:) + C(:) * D(:)
```

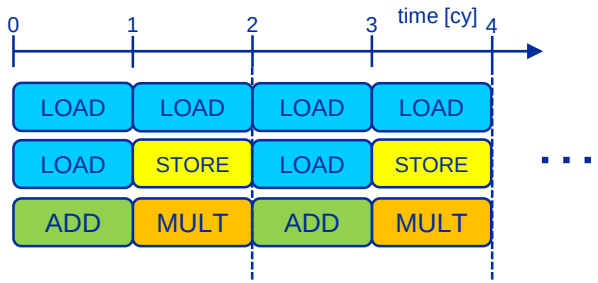
- Core limitations
  - AVX: 1 LD, ½ ST, 1 ADD, 1 MULT
  - SSE: 2 LD || (1 LD + 1 ST), 1 ADD, 1 MULT
  - Scalar: see SSE
- Performance limits
  - AVX: 3 cy for 4 iterations → $P_{\max} = \frac{8\ \text{flops}}{3\ \text{cy}} \times 2.2\ \frac{\text{Gcy}}{\text{s}} = 5.87\ \text{Gflop/s}$
    - Observed with Intel 17: 4.7 Gflop/s
  - SSE: 2 cy for 2 iterations → $P_{\max} = \frac{4\ \text{flops}}{2\ \text{cy}} \times 2.2\ \frac{\text{Gcy}}{\text{s}} = 4.4\ \text{Gflop/s}$
  - Scalar: 2 cy for 1 iteration → $P_{\max} = \frac{2\ \text{flops}}{2\ \text{cy}} \times 2.2\ \frac{\text{Gcy}}{\text{s}} = 2.2\ \text{Gflop/s}$

Assignment 2 – Task 2

Visualization of pipeline throughput (rotated compared to lecture)



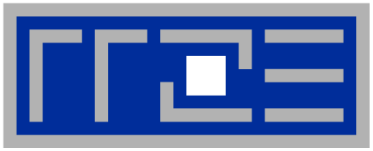
AVX: 2 x 4 flops in 3 cy



SSE: 2 x 2 flops in 2 cy
Scalar: 2 flops in 2 cy

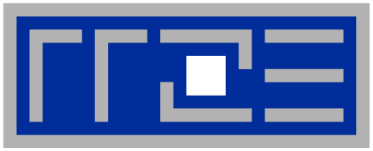
speedup = 2.67

Assignment 2 – Task 2



- **Conclusions**
  - Intel 17 with plain C code and standard options used AVX instructions
  - Gcc 4.8.5 might have used SSE or just inefficient AVX code. Only way to tell is to look at the assembly code.

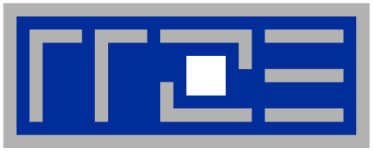
Assignment 2 – Task 3



```
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

Assignment 2 – Task 3



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

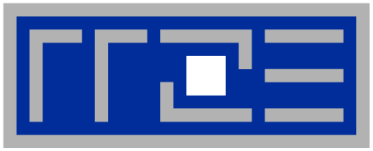
    1 AVX iteration == 2LD, 1 ADD, 1 MULT
    → 3 cycles/Iteration → 4·2/3 F/cy = 8/3 F/cy

  - 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

- SIMD vectorization w/ 2xAVX (8-way), no unrolling
  - Maximum throughput doubles

Assignment 2 – Task 3



- 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

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

- 4-way unrolling would change nothing.

Assignment 2 – Task 4

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