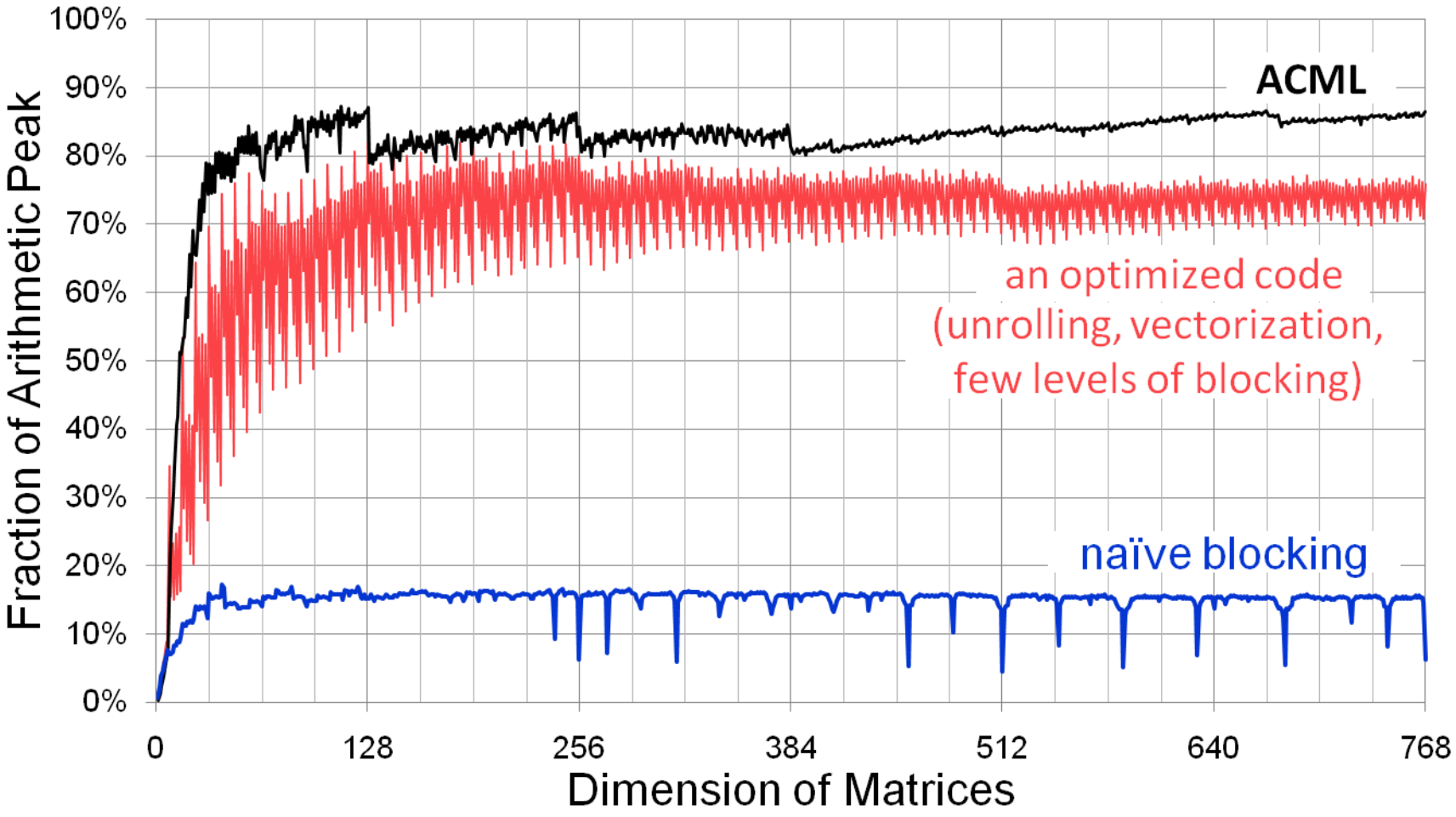
Notes on Homework 1

What is Peak ?



Summary of SSE intrinsics

#include <emmintrin.h>

Vector data type:

- **__m128d**

Load and store operations:

- **_mm_load_pd**
- **_mm_store_pd**
- **_mm_loadu_pd**
- **_mm_storeu_pd**

Load and broadcast across vector

- **_mm_load1_pd**

Arithmetic:

- **_mm_add_pd**
- **_mm_mul_pd**

2x2 Matrix Multiply

C_{00} += $A_{00}B_{10}$ + $A_{01}B_{00}$

C_{10} += $A_{10}B_{10}$ + $A_{11}B_{00}$

C_{01} += $A_{00}B_{11}$ + $A_{01}B_{01}$

C_{11} += $A_{10}B_{11}$ + $A_{11}B_{01}$

Rewrite as SIMD algebra

C00_C10 += A00_A10 * B10_B10

C01_C11 += A00_A10 * B11_B11

C00_C10 += A01_A11 * B00_B00

C01_C11 += A01_A11 * B01_B01

Example: multiplying 2x2 matrices

```
#include <emmintrin.h>
c1 = _mm_loadu_pd( C+0*lda );          //load unaligned block in C
c2 = _mm_loadu_pd( C+1*lda );
for( int i = 0; i < 2; i++ )
{
  a = _mm_load_pd( A+i*lda );//load aligned i-th column of A
  b1 = _mm_load1_pd( B+i+0*lda );      //load i-th row of B
  b2 = _mm_load1_pd( B+i+1*lda );
  c1=_mm_add_pd( c1, _mm_mul_pd( a, b1 ) ); //rank-1 update
  c2=_mm_add_pd( c2, _mm_mul_pd( a, b2 ) );
}
_mm_storeu_pd( C+0*lda, c1 );          //store unaligned block in C
_mm_storeu_pd( C+1*lda, c2 );
```

Other Issues

Checking efficiency of the compiler helps

- **Use -S option to see the generated assembly code**
- **Inner loop should consist mostly of ADDPD and MULPD ops, ADDSD and MULSD imply scalar computations**

- **Consider using another compiler**
  - **Options are PGI, PathScale and GNU**
  - **Cray C compiler doesn't seem to have any way to use SSE**
    - **It is capable of auto-vectorizing loops it recognizes**
- **Look through Goto and van de Geijn's paper**
- **Don't ignore the other optimizations**
  - **more compiler flags**
  - **loop unrolling**
  - **inlining**
  - **keywords (ask your classmates!)**