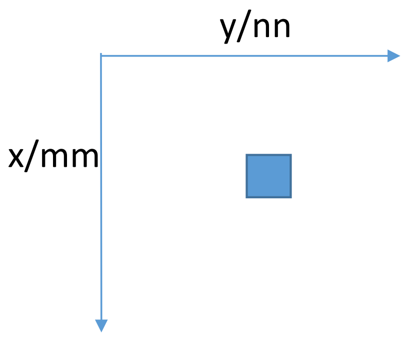
Lab 4

- Assigned on March 2
- Due midnight on March 20 (2.5 weeks)
- Goal: Add GPU capability to SUMMA to accelerate matrix multiplication
  - Implement your own GPU kernel:
    - A suggested method (need fixes, see the next slide)
    - In your report and presentation, describe how the algorithm works exactly
    - Next, can you make it faster? (after you make sure the above suggested method works)
      - Tune the thread block size? Tune the warp size? Tune the matrix block size?
        - Do they have to be identical?
      - Can you use 2-Dimensial thread blocks and Lab2's blocking algorithm?
  - Let your SUMMA program upload/dowload submatrix computations to one GPU.
  - Compare performance of your best GPU kernel to CUBLAS.
    - CUBLAS is the vendor BLAS library provided by Nvidia
  - Compare performance of your GPU-accelerated SUMMA with Cray LibSci ScaLapack.
  - http://www.soa-world.de/echelon/cuda-faq#measure-runtime-of-kernel

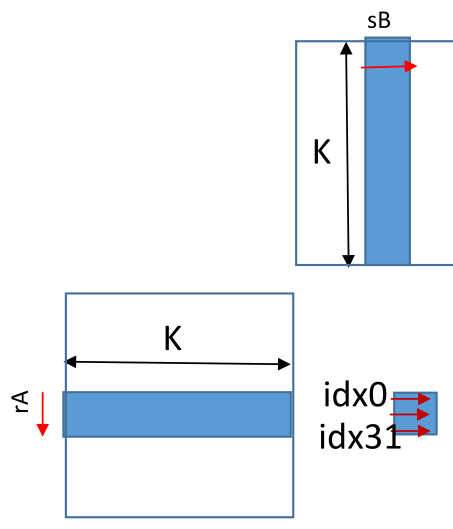
The code needs to be modified as needed.

- -32x32 matrix blocks
- -thread block size: 32
- -A, C are column major
- -B is row major
- -C -= A*B^T



```
#define WARP 32
__global__
void beastSgemm(
    int K,
    const float* __restrict__ dA, int lda,
    const float* __restrict__ dB, int ldb,
          float *dC,              int ldc)
{
    int n, k, kk;
    int mm = blockIdx.x*WARP;
    int nn = blockIdx.y*WARP;
    int idx = threadIdx.x;
    __shared__ float sB[WARP];
    float rA, rC[WARP];                // 16 lines
    #pragma unroll
    for (n = 0; n < WARP; n++)
        rC[n] = 0.0f;
    //Each thread computes 1 row.
    for (kk = 0; kk < K; kk+=WARP) {
        #pragma unroll
        for (k = 0; k < WARP; k++) {
            rA      = dA[(mm+idx)+(k+kk)*lda];
            sB[idx] = dB[(nn+idx)+(k+kk)*ldb];
            #pragma unroll
            for (n = 0; n < WARP; n++)
                rC[n] += rA*sB[n];
        }
    }
    #pragma unroll
    for (n = 0; n < WARP; n++)
        dC[(mm+idx)+(n+nn)*ldc] -= rC[n];
}

dim3 dimGrid(m/WARP, n/WARP);
dim3 dimBlock(WARP);
beastSgemm<<<dimGrid, dimBlock>>>(
    k,
    dA, ldm,
    dB, ldn,
    dD, ldm);
```



Each thread has 32 rC registers