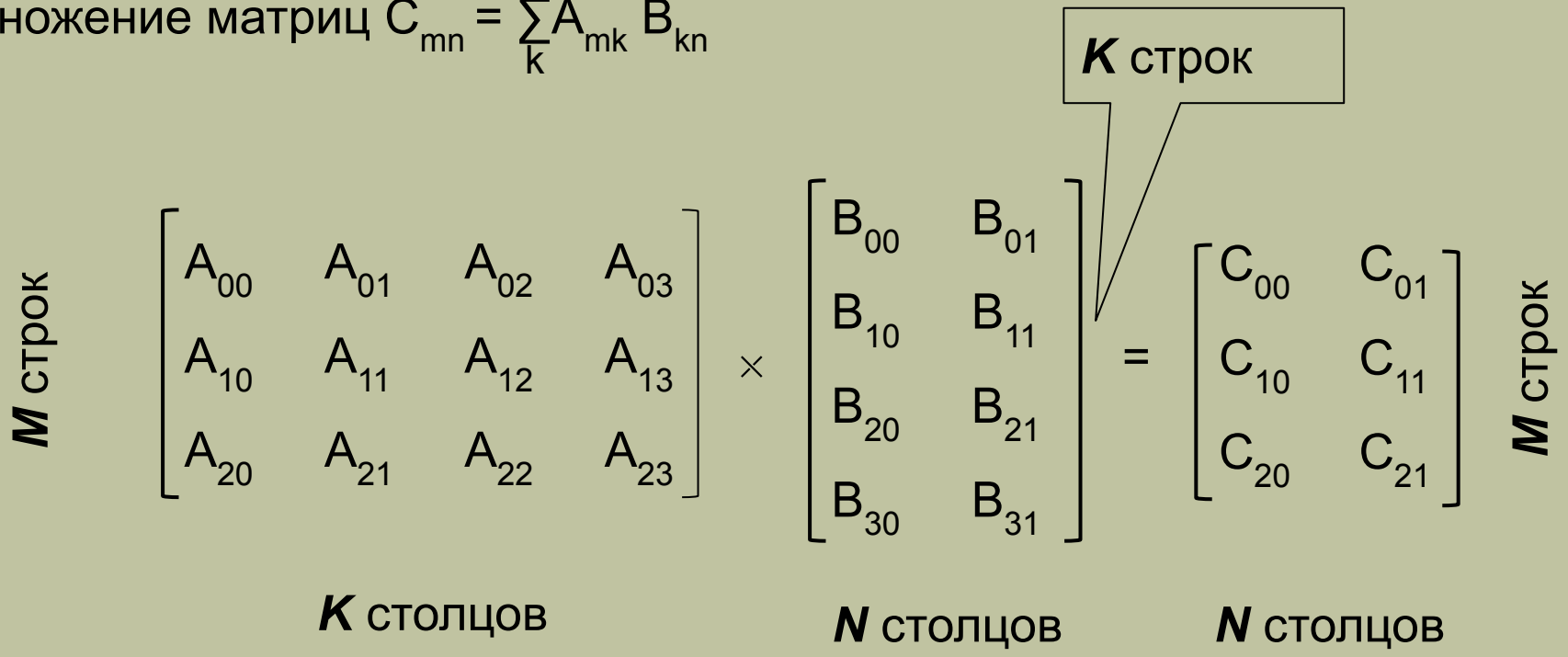
Лекция 13

Warp Matrix Functions (wmma)

Матричное произведение тензоров
Произведение матриц

Умножение матриц $C_{mn} = \sum_k A_{mk} B_{kn}$



```
#include <mma.h>
using namespace nvcuda;

__global__ void wmma_ker(half *a, half *b, float *c) {
   wmma::fragment<wmma::matrix_a, 16, 16, 16, half, wmma::col_major> a_frag;
   wmma::fragment<wmma::matrix_b, 16, 16, 16, half, wmma::row_major> b_frag;
   wmma::fragment<wmma::accumulator, 16, 16, 16, float> c_frag;

   wmma::fill_fragment(c_frag, 0.0f);
   wmma::load_matrix_sync(a_frag, a, 16);
   wmma::load_matrix_sync(b_frag, b, 16);

   wmma::mma_sync(c_frag, a_frag, b_frag, c_frag);

   wmma::store_matrix_sync(c, c_frag, 16, wmma::mem_row_major);
}
```

test_mm-0.cu

```
#include <stdio.h>
#include <malloc.h>
#include <mma.h>
using namespace nvcuda;

#define M 16
#define K 16
#define N 16

void hInit(half* A, half* B){
 for(int m=0;m<M;m++)
  for(int k=0;k<K;k++)
     A[k+m*K]=(half)((m+k*M)*0.01);
 for(int k=0;k<K;k++)
  for(int n=0;n<N;n++)
     B[n+k*N]=(half)1.0;
}
```

```
__global__ void wMMult(half *a, half *b, float *c) {
    wmma::fragment<wmma::matrix_a, 16, 16, 16, half, wmma::col_major> a_frag;
    wmma::fragment<wmma::matrix_b, 16, 16, 16, half, wmma::row_major> b_frag;
    wmma::fragment<wmma::accumulator, 16, 16, 16, float> c_frag;

    wmma::fill_fragment(c_frag, 0.0f);
    wmma::load_matrix_sync(a_frag, a, 16);
    wmma::load_matrix_sync(b_frag, b, 16);

    wmma::mma_sync(c_frag, a_frag, b_frag, c_frag);

    wmma::store_matrix_sync(c, c_frag, 16, wmma::mem_row_major);
}
```

```
int main(){
 half* A=(half*)calloc(M*K, sizeof(half));
 half* B=(half*)calloc(K*N, sizeof(half));
 float* C=(float*)calloc(M*N, sizeof(float));

 hInit(A,B);

 half *Ad, *Bd;
 float *Cd;
 cudaMalloc((void**)&Ad,  M*K*sizeof(half));
 cudaMalloc((void**)&Bd,  K*N*sizeof(half));
 cudaMalloc((void**)&Cd,  M*N*sizeof(float));
```

```
cudaMemcpy(Ad, A, M*K*sizeof(half), cudaMemcpyHostToDevice);
cudaMemcpy(Bd, B, M*K*sizeof(half), cudaMemcpyHostToDevice);
cudaMemcpy(Cd, C, M*K*sizeof(float), cudaMemcpyHostToDevice);
cudaMemset(Cd, 0, M*N*sizeof(float));

wMMult<<<1,32>>>(Ad,Bd,Cd);
cudaDeviceSynchronize();
```

```
 cudaMemcpy(C, Cd, M*K*sizeof(float), cudaMemcpyDeviceToHost);
 for(int m=0;m<M;m+=M/8){
  for(int n=0;n<N;n+=N/8)
   printf("%g\t", C[n+m*N]);
  printf("\n");
  }

  free(A);
  free(B);
  free(C);

  cudaFree(Ad);
  cudaFree(Bd);
  cudaFree(Cd);

  return 0;
}
```

```
Lecture6/Lab6> nvcc  -arch=sm_75   test_mm-0.cu  -o test_mm-0
Lecture6/Lab6> test_mm-0

1.20005 1.20005 1.20005 1.20005 1.20005 1.20005 1.20005 1.20005
6.31982 6.31982 6.31982 6.31982 6.31982 6.31982 6.31982 6.31982
11.4404 11.4404 11.4404 11.4404 11.4404 11.4404 11.4404 11.4404
16.5596 16.5596 16.5596 16.5596 16.5596 16.5596 16.5596 16.5596
21.6797 21.6797 21.6797 21.6797 21.6797 21.6797 21.6797 21.6797
26.8008 26.8008 26.8008 26.8008 26.8008 26.8008 26.8008 26.8008
31.9199 31.9199 31.9199 31.9199 31.9199 31.9199 31.9199 31.9199
37.0391 37.0391 37.0391 37.0391 37.0391 37.0391 37.0391 37.0391
```

test_mm.cu

```
#include <stdio.h>
#include <malloc.h>
#define M 32
#define K 32
#define N 32
#define Kb 16

void hInit(float* A, float* B){
 for(int m=0;m<M;m++)
  for(int k=0;k<K;k++)
    A[k+m*K]=(float)(k+m*K);

 for(int k=0;k<K;k++)
  for(int n=0;n<N;n++)
    B[n+k*N]=(float)1.0;
}
```

```
void hMMult(float* A, float* B, float* C){
 for(int m=0;m<M;m++)
  for(int n=0;n<N;n++){
    float acc=0.0;
    for(int tk=0;tk<K/Kb;tk++)
     for(int kb=0; kb<Kb; kb++)
      acc+=A[kb+tk*Kb+m*K]*B[n+(kb+tk*Kb)*N];
    C[n+m*N]=acc;
  }
}
```

```
int main(){
 float* A=(float*)calloc(M*K, sizeof(float));
 float* B=(float*)calloc(K*N, sizeof(float));
 float* C=(float*)calloc(M*N, sizeof(float));

 hInit(A,B);
 hMMult(A, B, C);

 for(int m=0;m<M;m+=M/8){
  for(int n=0;n<N;n+=N/8)
   printf("%g\t", C[n+m*N]);
  printf("\n");
 }

 free(A);  free(B); free(C);
 return 0;
}
```

```
/Lecture6/Lab6> nvcc test_mm.cu -o test_mm
/Lecture6/Lab6> ./test_mm


496     496     496     496     496     496     496     496
4592    4592    4592    4592    4592    4592    4592    4592
8688    8688    8688    8688    8688    8688    8688    8688
12784   12784   12784   12784   12784   12784   12784   12784
16880   16880   16880   16880   16880   16880   16880   16880
20976   20976   20976   20976   20976   20976   20976   20976
25072   25072   25072   25072   25072   25072   25072   25072
29168   29168   29168   29168   29168   29168   29168   29168
```

```
void hMMult(float* A, float* B, float* C){
 for(int tm=0;tm<M/Mb;tm++)
 for(int tn=0;tn<N/Nb;tn++)
 for(int tk=0;tk<K/Kb;tk++){
  for(int mb=0;mb<Mb;mb++)
   for(int nb=0;nb<Nb;nb++)
     for(int kb=0; kb<Kb; kb++)
      C[nb+tn*Nb+(mb+tm*Mb)*N]+=
      A[kb+tk*Kb+(mb+tm*Mb)*K]*
      B[nb+tn*Nb+(kb+tk*Kb)*N];
   }
}
```

Из test_mm-1.cu

```
void hMMult(float* A, float* B, float* C){
 for(int tm=0;tm<M/Mb;tm++)
  for(int tn=0;tn<N/Nb;tn++)
   for(int tk=0;tk<K/Kb;tk++){
       for(int mb=0;mb<Mb;mb++)
        for(int nb=0;nb<Nb;nb++)
         for(int kb=0; kb<Kb; kb++)
           (C+tn*Nb+tm*Mb*N)[nb+mb*N]+=
           (A+tk*Kb+tm*Mb*K)[kb+mb*K]*
           (B+tn*Nb+tk*Kb*N)[nb+kb*N];
   }
}
```

Из test_mm-2.cu

```
void hMMult(float* A, float* B, float* C){
 for(int mb=0;mb<Mb;mb++)
   for(int nb=0;nb<Nb;nb++)
     for(int kb=0; kb<Kb; kb++)
       C[nb+mb*N]+= A[kb+mb*K]*B[nb+kb*N];
}

Int main(){
……….
 for(int tm=0;tm<M/Mb;tm++)
  for(int tn=0;tn<N/Nb;tn++)
   for(int tk=0;tk<K/Kb;tk++)
     hMMult(A+tk*Kb+tm*Mb*K,  B+tn*Nb+tk*Kb*N,  C+tn*Nb+tm*Mb*N);
……..
```

Из test_mm-3.cu

```
Lecture6/Lab6> nvcc  test_mm-3.cu  -o test_mm-3
Lecture6/Lab6> ./test_mm-3

496     496     496     496     496     496     496     496
4592    4592    4592    4592    4592    4592    4592    4592
8688    8688    8688    8688    8688    8688    8688    8688
12784   12784   12784   12784   12784   12784   12784   12784
16880   16880   16880   16880   16880   16880   16880   16880
20976   20976   20976   20976   20976   20976   20976   20976
25072   25072   25072   25072   25072   25072   25072   25072
29168   29168   29168   29168   29168   29168   29168   29168
```

test_mm-6.cu

```
#include <stdio.h>
#include <malloc.h>
#include <mma.h>
using namespace nvcuda;

#define M 32
#define K 32
#define N 32

#define Kb 16
#define Mb 16
#define Nb 16
```

```
void hInit(half* A, half* B){
 for(int m=0;m<M;m++)
  for(int k=0;k<K;k++)
     A[k+m*K]=(half)((m+k*M)*0.01);

 for(int k=0;k<K;k++)
  for(int n=0;n<N;n++)
     B[n+k*N]=(half)1.0;
}
```

```
__global__ void wMMult(half *a, half *b, float *c) {
   int Mw = (blockIdx.x * blockDim.x + threadIdx.x)/warpSize;
   int Nw = (blockIdx.y * blockDim.y + threadIdx.y);

   wmma::fragment<wmma::matrix_a, Mb, Nb, Kb, half, wmma::col_major> a_frag;
   wmma::fragment<wmma::matrix_b, Mb, Nb, Kb, half, wmma::col_major> b_frag;
   wmma::fragment<wmma::accumulator, Mb, Nb, Kb, float> c_frag;

   wmma::fill_fragment(c_frag, 0.0f);
```

```
////////////////////////////////////////////////////////////////

  for(int tk=0; tk<K; tk+=Kb){
    int a_row=Mw*Mb;
    int a_col=tk;
    int b_row =tk;
    int b_col=Nw*Nb;

    wmma::load_matrix_sync(a_frag, a+a_row+a_col*M, M);
    wmma::load_matrix_sync(b_frag, b+b_row+b_col*K, K);
    wmma::mma_sync(c_frag, a_frag, b_frag, c_frag);
  }
  int c_row=Mw*Mb;
  int c_col=Nw*Nb;
  wmma::store_matrix_sync(c+c_row+c_col*M, c_frag, M, wmma::mem_col_major);

////////////////////////////////////////////////////////////////////////
}
```

```
int main(){
 half* A=(half*)calloc(M*K, sizeof(half));
 half* B=(half*)calloc(K*N, sizeof(half));
 float* C=(float*)calloc(M*N, sizeof(float));

 hInit(A,B);

 half *Ad, *Bd;
 float *Cd;
 cudaMalloc((void**)&Ad,  M*K*sizeof(half));
 cudaMalloc((void**)&Bd,  K*N*sizeof(half));
 cudaMalloc((void**)&Cd,  M*N*sizeof(float));

 cudaMemcpy(Ad, A, M*K*sizeof(half), cudaMemcpyHostToDevice);
 cudaMemcpy(Bd, B, M*K*sizeof(half), cudaMemcpyHostToDevice);
  cudaMemset(Cd, 0, M*N*sizeof(float));
```

```
wMMult<<<dim3(2,2),dim3(32,1)>>>(Ad,Bd,Cd);
cudaDeviceSynchronize();

cudaMemcpy(C, Cd, M*K*sizeof(float), cudaMemcpyDeviceToHost);
for(int m=0;m<M;m+=M/8){
 for(int n=0;n<N;n+=N/8)
  printf("%g\t", C[n+m*N]);
 printf("\n");
 }
```

```
  free(A);
  free(B);
  free(C);

  cudaFree(Ad);
  cudaFree(Bd);
  cudaFree(Cd);

  return 0;
}
```

```
Lecture6/Lab6> nvcc  -arch=sm_75   test_mm-6.cu   -o test_mm-6
Lecture6/Lab6> ./test_mm-6

4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
4.96018 45.9199 86.8789 127.84  168.801 209.762 250.719 291.68
```