4 B

```
#include <cuda_runtime.h>
#include <iostream>
_global___void matmul(int* A, int* B, int* C, int N) {
int Row = blockIdx.y*blockDim.y+threadIdx.y;
int Col = blockIdx.x*blockDim.x+threadIdx.x;
if (Row < N && Col < N) {
int Pvalue = 0;
for (int k = 0; k < N; k++) {
Pvalue += A[Row*N+k] * B[k*N+Col];
}
C[Row*N+Col] = Pvalue;
}
}
int main() {
int N = 512;
int size = N * N * sizeof(int);
int* A, * B, * C;
int* dev_A, * dev_B, * dev_C;
cudaMallocHost(&A, size);
cudaMallocHost(&B, size);
cudaMallocHost(&C, size);
cudaMalloc(&dev_A, size);
cudaMalloc(&dev_B, size);
cudaMalloc(&dev_C, size);
// Initialize matrices A and B
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
A[i*N+j] = i*N+j;
B[i*N+j] = j*N+i;
}
}
cudaMemcpy(dev_A, A, size,
cudaMemcpyHostToDevice);
cudaMemcpy(dev_B, B, size,
cudaMemcpyHostToDevice);
dim3 dimBlock(16, 16);
dim3 dimGrid(N/dimBlock.x, N/dimBlock.y);
matmul<<<dimGrid, dimBlock>>>(dev_A, dev_B,
dev_C, N);
cudaMemcpy(C, dev_C
// Print the result
for (int i = 0; i < 10; i++) {
for (int j = 0; j < 10; j++) {
std::cout << C[i*N+j] << " ";
}
std::cout << std::endl;
}
// Free memory
cudaFree(dev_A);
cudaFree(dev_B);
cudaFree(dev_C);
cudaFreeHost(A);
cudaFreeHost(B);
cudaFreeHost(C);
return 0;
```