

```

1  #define GRID_SIZE 256
2  #define BLOCK_SIZE 128
3
4  __global__
5  void gpu_dotproduct_stage1(const double *gpu_x, const double *gpu_y, size_t size,
6  double *gpu_result_stage1){
7
8      size_t thread_id_global = blockIdx.x*blockDim.x + threadIdx.x;
9      __shared__ double shared_m[BLOCK_SIZE];
10
11      // I think, this is the right way:
12      double thread_dp = 0;
13      for (unsigned int i = thread_id_global; i<size; i += blockDim.x * gridDim.x)
14          thread_dp += gpu_x[i] * gpu_y[i];
15      shared_m[threadIdx.x] = thread_dp;
16
17      // now the reduction
18      for(int stride = blockDim.x/2; stride>0; stride/=2){
19          __syncthreads();
20          if (threadIdx.x < stride){
21              shared_m[threadIdx.x] += shared_m[threadIdx.x + stride];
22          }
23      }
24
25      // thread 0 writes result
26      if (threadIdx.x == 0){
27          gpu_result_stage1[blockIdx.x] = shared_m[0];
28      }
29  }
30
31  __global__
32  void gpu_dotproduct_stage2(double *gpu_result_stage1, double *gpu_result_stage2){
33
34      // only one block has a job here
35      if (blockIdx.x == 0){
36          //size_t thread_id_global = blockIdx.x*blockDim.x + threadIdx.x;
37          __shared__ double shared_m[BLOCK_SIZE];
38
39          // this time, the lecture is correct
40          double thread_sum = 0;
41          for (unsigned int i = threadIdx.x; i<GRID_SIZE; i += blockDim.x)
42              thread_sum += gpu_result_stage1[i];
43          shared_m[threadIdx.x] = thread_sum;
44
45          // now the reduction
46          for(int stride = blockDim.x/2; stride>0; stride/=2){
47              __syncthreads();
48              if (threadIdx.x < stride){
49                  shared_m[threadIdx.x] += shared_m[threadIdx.x + stride];
50              }
51          }
52          // thread 0 writes result
53          if (threadIdx.x == 0){
54              //printf("hi, im thread 0 and im now writing %1.5e\n", shared_m[0]);
55              *gpu_result_stage2 = shared_m[0];
56          }
57      }
58  }
59
60
61  __global__
62  void gpu_dotproduct_atomicAdd(const double *gpu_x, const double *gpu_y, size_t size,
63  double *gpu_result){
64
65      size_t thread_id_global = blockIdx.x*blockDim.x + threadIdx.x;
66      if (thread_id_global == 0)
67          *gpu_result = 0;
68
69      __shared__ double shared_m[BLOCK_SIZE];
70
71      // I think, this is the right way:
72      double thread_dp = 0;

```

```

72     for (unsigned int i = thread_id_global; i<size; i += blockDim.x * gridDim.x)
73         thread_dp += gpu_x[i] * gpu_y[i];
74     shared_m[threadIdx.x] = thread_dp;
75
76
77     // now the reduction
78     for(int stride = blockDim.x/2; stride>0; stride/=2){
79         __syncthreads();
80         if (threadIdx.x < stride){
81             shared_m[threadIdx.x] += shared_m[threadIdx.x + stride];
82         }
83     }
84
85     __syncthreads();
86     // thread 0 writes result
87     if (threadIdx.x == 0){
88         atomicAdd(gpu_result, shared_m[0]);
89         //gpu_result_stage1[blockIdx.x] = shared_m[0];
90     }
91 }
92

```