

Assignment # 03


Task # 1:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~ $ ls  
cuda  
cdc-p176075@lmar ~ $ ls -lh  
total 4.0K  
drwxr-xr-x 2 cdc-p176075 cdc-p176075 4.0K 11:33 25 جون cuda  
cdc-p176075@lmar ~ $
```

Task # 2:

```
cdc-p176075@lmar:~/cuda  
cdc-p176075@lmar ~ $ ls  
cuda  
cdc-p176075@lmar ~ $ ls -lh  
total 4.0K  
drwxr-xr-x 2 cdc-p176075 cdc-p176075 4.0K 11:33 25 جون cuda  
cdc-p176075@lmar ~ $ cd cuda/  
cdc-p176075@lmar ~/cuda $ ls  
hello.cu  
cdc-p176075@lmar ~/cuda $ ls -lh  
total 4.0K  
-rw-r--r-- 1 cdc-p176075 cdc-p176075 106 11:33 25 جون hello.cu  
cdc-p176075@lmar ~/cuda $ nvcc hello.cu -o hello  
cdc-p176075@lmar ~/cuda $ ./hello  
Hello World!  
cdc-p176075@lmar ~/cuda $
```

Task # 3:

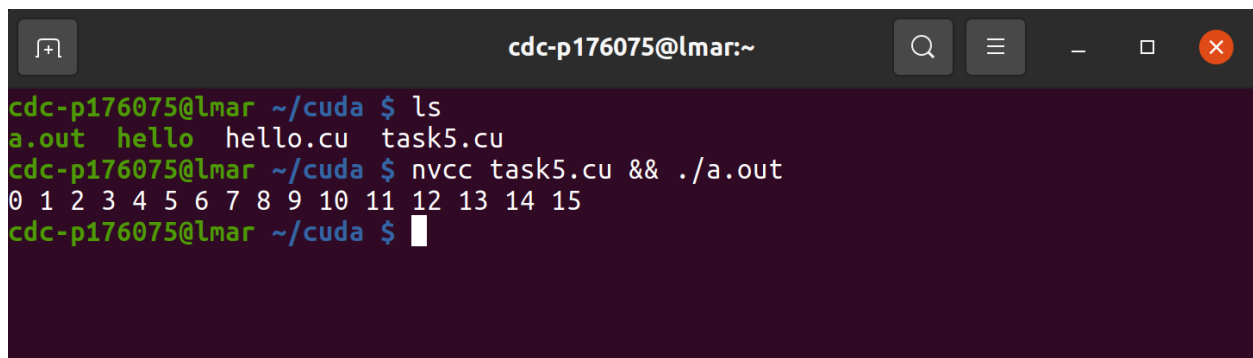


```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc hello.cu -o hello
cdc-p176075@lmar ~/cuda $ ./hello
Hello World!
cdc-p176075@lmar ~/cuda $
```

The screenshot shows a terminal window with the title bar "cdc-p176075@lmar:~/cuda". The user has compiled a CUDA file "hello.cu" using "nvcc" and executed it, resulting in the output "Hello World!". On the right side of the terminal, the source code for "hello.cu" is displayed, showing a simple C program that prints "Hello World!".

Task # 4: I have done this, and it helps me alot in case of connection loss.

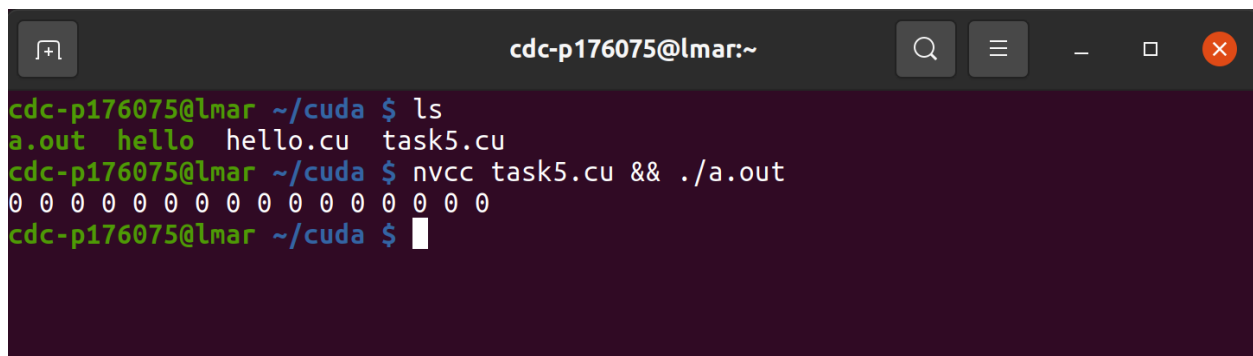
Task # 5a:



```
cdc-p176075@lmar:~
cdc-p176075@lmar ~/cuda $ ls
a.out  hello  hello.cu  task5.cu
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cdc-p176075@lmar ~/cuda $
```

The screenshot shows a terminal window with the title bar "cdc-p176075@lmar:~". The user has listed the contents of the directory, which include "a.out", "hello", "hello.cu", and "task5.cu". They then compiled "task5.cu" using "nvcc" and executed the resulting "a.out" file, which outputs a sequence of numbers from 0 to 15.

Task # 5b:



```
cdc-p176075@lmar:~
cdc-p176075@lmar ~/cuda $ ls
a.out  hello  hello.cu  task5.cu
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
cdc-p176075@lmar ~/cuda $
```

The screenshot shows a terminal window with the title bar "cdc-p176075@lmar:~". The user has listed the contents of the directory, which include "a.out", "hello", "hello.cu", and "task5.cu". They then compiled "task5.cu" using "nvcc" and executed the resulting "a.out" file, which outputs a sequence of zeros.

Task # 5c:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~/cuda $ ls  
a.out hello hello.cu task5.cu  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
cdc-p176075@lmar ~/cuda $
```

Task # 5d:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
cdc-p176075@lmar ~/cuda $
```

Task # 5e:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0  
cdc-p176075@lmar ~/cuda $
```

Task # 5f:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
0 0 0 0 0 0 0 0 0 1 2 3 4 5 6 7  
cdc-p176075@lmar ~/cuda $
```

Task # 5g:

```
cdc-p176075@lmar:~  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888  
cdc-p176075@lmar ~/cuda $
```

Task # 5h:

```
task5.cu 2 x  
task5.cu > myHelloOnGPU(int *)  
1  /* task-5.cu */  
2  #include <stdio.h>  
3  
4  
5  global void myHelloOnGPU(int *array){  
6      // Position 1  
7      array[blockIdx.x * 2 + gridDim.x * threadIdx.x] = 111 * (blockIdx.x + 1);  
8  }  
9  
10  
11  int main(){  
12      int N = 16;  
13      int *cpuArray = (int*)malloc(sizeof(int)*N);  
14      int *gpuArray;  
15      cudaMalloc((void **)&gpuArray, sizeof(int)*N);  
16  
17      // Position 2  
18      myHelloOnGPU<<<N/2, 1>>>(gpuArray);  
19      cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);  
20  
21      for(int i=0; i<N; i++){  
22          printf("%d ", cpuArray[i]);  
23      }  
24      printf("\n");  
25      return 0;  
26  }  
27  
28  cdc-p176075@lmar:~/cuda  
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out  
111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0  
cdc-p176075@lmar ~/cuda $
```

Task # 5j:

```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
cdc-p176075@lmar ~/cuda $
```

Task # 5k:

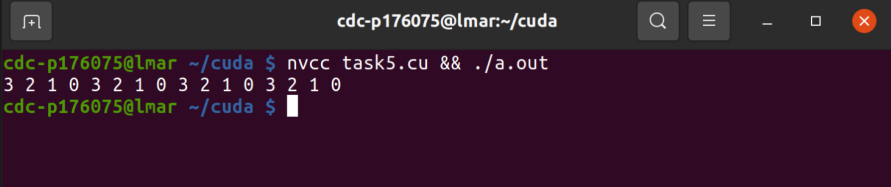
```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0
cdc-p176075@lmar ~/cuda $
```

Task # 5m:

```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444
cdc-p176075@lmar ~/cuda $
```

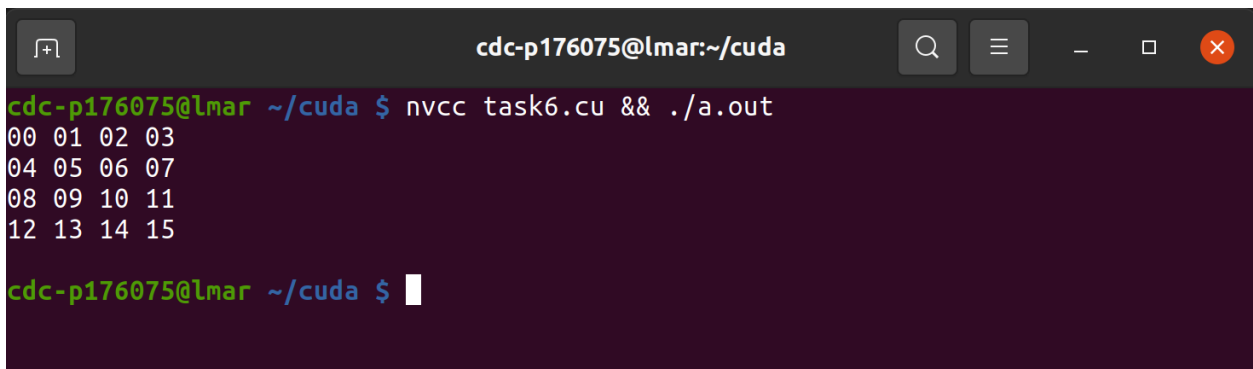
Task # 5n:

```
task5.cu > myHelloOnGPU(int *)
1  /* task-5.cu */
2  #include <stdio.h>
3
4
5  global void myHelloOnGPU(int *array){
6      // Position 1
7      array[blockIdx.x * blockDim.x + threadIdx.x] = [ blockDim.x - threadIdx.x - 1];
8
9  }
10
11 int main(){
12     int N = 16;
13     int *cpuArray = (int*)malloc(sizeof(int)*N);
14     int *gpuArray;
15     cudaMalloc((void **)&gpuArray, sizeof(int)*N);
16
17     // Position 2
18     myHelloOnGPU<<<N/4, N/4>>>(gpuArray);
19     cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
20
21     for(int i=0; i<N; i++){
22         printf("%d ", cpuArray[i]);
23     }
24     printf("\n");
25     return 0;
26 }
27
28
```



```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task5.cu && ./a.out
3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
cdc-p176075@lmar ~/cuda $
```

Task # 6a:



```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15

cdc-p176075@lmar ~/cuda $
```

Task # 6b:

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar:~/cuda $
```

Task # 6c:

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15

cdc-p176075@lmar:~/cuda $
```

Task # 6d:

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar:~/cuda $
```

Task # 6e:

```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00

cdc-p176075@lmar ~/cuda $
```

Task # 6f:

```
task6.cu > myHelloOnGPU(int *)
2  * name: task-6.cu
3  */
4
5  #include<stdio.h>
6  global void myHelloOnGPU(int *array){
7  // Position-1
8  array[blockIdx.x * gridDim.x + blockIdx.x] = 11 * (blockIdx.x + 1);
9  }
10
11 int main(){
12     int N = 16;
13     int *cpuArray = (int*)malloc(sizeof(int)*N);
14     int *gpuArray;
15     cudaMalloc((void **)&gpuArray, sizeof(int)*N);
16     // Position-2
17     dim3 dimGrid(N/4, 1, 1);
18     dim3 dimBlock(1, 1, 1);
19
20     myHelloOnGPU<<<dimGrid, dimBlock>>>(gpuArray);
21
22     cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
23     for (int i = 0; i < N/4; i++){
24         for (int j = 0; j < N/4; j++){
25             printf("%.2d ", cpuArray[i*N/4+j]);
26         }
27         printf("\n");
28     }
29     printf("\n");
30     return 0;
31 }
32
33
34
```

```
cdc-p176075@lmar:~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
00 22 00 00
00 00 33 00
00 00 00 44

cdc-p176075@lmar ~/cuda $
```


Task # 6g(1):

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
00 00 00 00
00 00 00 00
00 00 00 00

cdc-p176075@lmar:~/cuda $
```

Task # 6h(1):

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00

cdc-p176075@lmar:~/cuda $
```

Task # 6g(2):

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44

cdc-p176075@lmar:~/cuda $
```

Task # 6h(2):

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
00 00 00 11
00 00 00 22
00 00 00 33
00 00 00 44

cdc-p176075@lmar:~/cuda $
```

Task # 6j:

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 22 33 44
11 22 33 44
11 22 33 44
11 22 33 44

cdc-p176075@lmar:~/cuda $
```

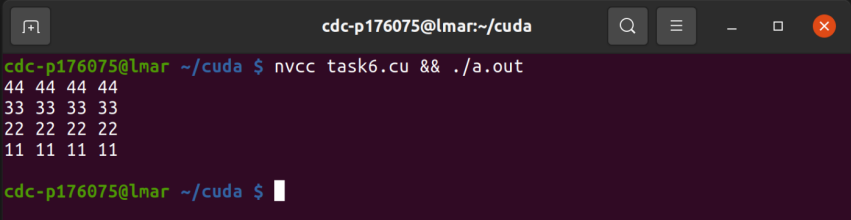
Task # 6k:

```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 11 11 11
22 22 22 22
33 33 33 33
44 44 44 44

cdc-p176075@lmar:~/cuda $
```

Task # 6m:

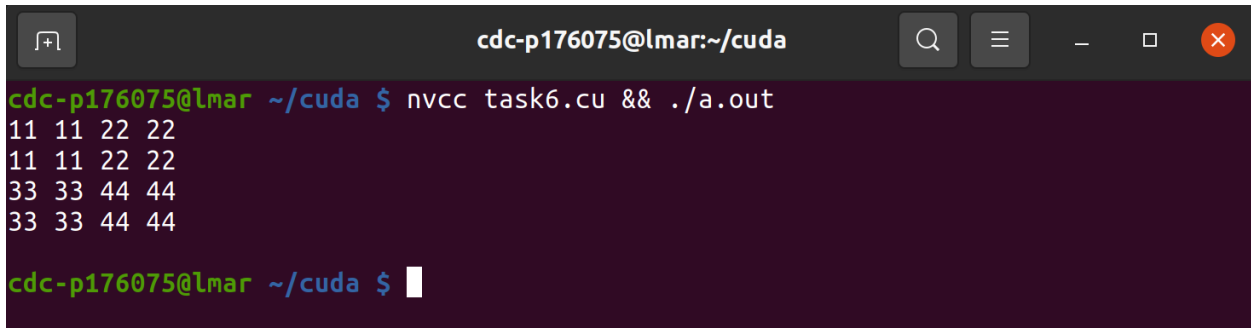
```
task6.cu > myHelloOnGPU(int *)
2  * name: task-6.cu
3  */
4
5  #include<stdio.h>
6  global void myHelloOnGPU(int *array){
7  // Position-1
8  array[blockIdx.x * blockDim.x + threadIdx.x] = 11*[(blockDim.x - blockIdx.x)];
9  }
10
11 int main(){
12     int N = 16;
13     int *cpuArray = (int*)malloc(sizeof(int)*N);
14     int *gpuArray;
15     cudaMalloc((void **)&gpuArray, sizeof(int)*N);
16     // Position-2
17     dim3 dimGrid(N/4, 1, 1);
18     dim3 dimBlock(N/4, 1, 1);
19
20     myHelloOnGPU(<<dimGrid, dimBlock>>>(gpuArray);
21
22     cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
23     for (int i = 0; i < N/4; i++){
24         for (int j = 0; j < N/4; j++){
25             printf("%.2d ", cpuArray[i*N/4+j]);
26         }
27         printf("\n");
28     }
29     printf("\n");
30     return 0;
31 }
32
33
34
```



```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
44 44 44 44
33 33 33 33
22 22 22 22
11 11 11 11

cdc-p176075@lmar ~/cuda $
```

Task # 6n:



```
cdc-p176075@lmar:~/cuda $ nvcc task6.cu && ./a.out
11 11 22 22
11 11 22 22
33 33 44 44
33 33 44 44

cdc-p176075@lmar ~/cuda $
```

Task # 6o:

```
task6.cu > main()
1  /*
2  * name: task-6.cu
3  */
4
5  #include<stdio.h>
6  global void myHelloOnGPU(int *array){
7      // Position-1
8      int index_x = blockIdx.x * blockDim.x + threadIdx.x;
9      int index_y = blockIdx.y * blockDim.y + threadIdx.y;
10     array[index_y * blockDim.x * blockDim.y + index_x] =
11     11 * (( blockDim.x * gridDim.x )-(( blockIdx.x * gridDim.x - blockDim.x*1 )+ ( blockIdx.y* gridDim.x )));
12 }
13
14 int main(){
15     int N = 16;
16     int *cpuArray = (int*)malloc(sizeof(int)*N);
17     int *gpuArray;
18     cudaMalloc((void **)&gpuArray, sizeof(int)*N);
19     // Position-2
20     dim3 dimGrid(N/8, N/8, 1);
21     dim3 dimBlock(N/8, N/8, 1);
22
23     myHelloOnGPU<<<dimGrid, dimBlock>>>(gpuArray);
24
25     cudaMemcpy(cpuArray, gpuArray, sizeof(int)*N, cudaMemcpyDeviceToHost);
26     for (int i = 0; i < N/4; i++){
27         for (int j = 0; j < N/4; j++){
28             printf("%2.2d ", cpuArray[i*N/4+j]);
29         }
30         printf("\n");
31     }
32     printf("\n");
33     return 0;
34 }
35
36
37
```

cdc-p176075@lmar:~/cuda

cdc-p176075@lmar ~/cuda \$ nvcc task6.cu && ./a.out

44 44 33 33
44 44 33 33
22 22 11 11
22 22 11 11

cdc-p176075@lmar ~/cuda \$

Task # 7:

```
#include <stdlib.h>
global void add(int *a, int *b, int *c) {
    // Position 1: To write Code here later
    int n = 16;
    int index = blockIdx.x * blockDim.x + threadIdx.x ;
    int stride = gridDim.x * blockDim.x;
    for (int i = index; i < n; i+=stride)
        c[i] = a[i] + b[i];
}

int main()
{
    int *a, *b, *c, *da, *db, *dc, N=16, i;
    a = (int*)malloc(sizeof(int)*N); // allocate host mem
    b = (int*)malloc(sizeof(int)*N); // and assign random
    c = (int*)malloc(sizeof(int)*N); // memory
    // Write code to initialize both a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = b[i] = 1;
    }
    cudaMalloc((void **)&da, sizeof(int)*N);
    cudaMalloc((void **)&db, sizeof(int)*N);
    cudaMalloc((void **)&dc, sizeof(int)*N);
    cudaMemcpy(da, a, sizeof(int)*N, cudaMemcpyHostToDevice);
    cudaMemcpy(db, b, sizeof(int)*N, cudaMemcpyHostToDevice);
    dim3 dimGrid(N/8, 1, 1);
    dim3 dimBlock(N/4, 1, 1);

    add<<<dimGrid,dimBlock>>>(da, db, dc);

    cudaMemcpy(c, dc, sizeof(int)*N, cudaMemcpyDeviceToHost);
    for (i = 0; i < N; i++) {
        printf("a[%d] + b[%d] = %d\n", i, i, c[i]);
    }
}
```

cdc-p176075@lmar:~/cuda

cdc-p176075@lmar ~/cuda \$ nvcc task7.cu && ./a.out

```
a[0] + b[0] = 2
a[1] + b[1] = 2
a[2] + b[2] = 2
a[3] + b[3] = 2
a[4] + b[4] = 2
a[5] + b[5] = 2
a[6] + b[6] = 2
a[7] + b[7] = 2
a[8] + b[8] = 2
a[9] + b[9] = 2
a[10] + b[10] = 2
a[11] + b[11] = 2
a[12] + b[12] = 2
a[13] + b[13] = 2
a[14] + b[14] = 2
a[15] + b[15] = 2
```

cdc-p176075@lmar ~/cuda \$

Task # 8:

```
task8.cu > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  global void add(int *a, int *b, int *c) {
4  // Position 1: To write Code here later
5  int Ix, Iy, index;
6  int n = 16;
7  Ix = blockIdx.x * blockDim.x + threadIdx.x;
8  Iy = blockIdx.y * blockDim.y + threadIdx.y;
9  index = Ix * blockDim.x * gridDim.y + Iy * blockDim.y * gridDim.y ;
10 int stride = 1 ;
11 for (int i = index; i < n; i+=stride)
12     c[i] = a[i] + b[i];
13 }
14 int main()
15 {
16     int *a, *b, *c, *da, *db, *dc, N=16, i, j;
17     a = (int*)malloc(sizeof(int)*N); // allocate host mem
18     b = (int*)malloc(sizeof(int)*N); // and assign random
19     c = (int*)malloc(sizeof(int)*N); // memory
20     // Write code to initialize both a and b to 1's.
21     for (i = 0; i < N; i++) {
22         a[i] = b[i] = 1;
23     }
24     cudaMalloc((void **)&da, sizeof(int)*N);
25     cudaMalloc((void **)&db, sizeof(int)*N);
26     cudaMalloc((void **)&dc, sizeof(int)*N);
27     cudaMemcpy(da, a, sizeof(int)*N, cudaMemcpyHostToDevice);
28     cudaMemcpy(db, b, sizeof(int)*N, cudaMemcpyHostToDevice);
29     dim3 dimGrid(N/8, N/8, 1);
30     dim3 dimBlock(N/8, N/8, 1);
31     add<<<dimGrid,dimBlock>>>>(da, db, dc);
32     cudaMemcpy(c, dc, sizeof(int)*N, cudaMemcpyDeviceToHost);
33     for (j = 0; j < N/4; j++) {
34         for (i = 0; i < N/4; i++) {
35             printf("a[%d] + b[%d] = %d\n", j*N/4+i, j*N/4+i, c[j*N/4+i]);
36         }
37         printf("\n");
38     }
39 }
```

```
cdc-p176075@lmar ~/cuda
cdc-p176075@lmar ~/cuda $ nvcc task8.cu && ./a.out
a[0] + b[0] = 2
a[1] + b[1] = 2
a[2] + b[2] = 2
a[3] + b[3] = 2
a[4] + b[4] = 2
a[5] + b[5] = 2
a[6] + b[6] = 2
a[7] + b[7] = 2
a[8] + b[8] = 2
a[9] + b[9] = 2
a[10] + b[10] = 2
a[11] + b[11] = 2
a[12] + b[12] = 2
a[13] + b[13] = 2
a[14] + b[14] = 2
a[15] + b[15] = 2
cdc-p176075@lmar ~/cuda $
```

Ln 2, Col 20 Spaces: 4 UTF-8 LF CUDA C++ kite: unsupported Linux