

Practice

- `cudaDeviceSynchronize()`; checks **if** there's some process in kernel to wait for, else it continues normally with host lines.

Lab 07

Code a program in c/c++ using CUDA in which you implement a kernel that inverts the order of the elements of an integer vector filled randomly, and that saves the values in another vector considering the requirements:

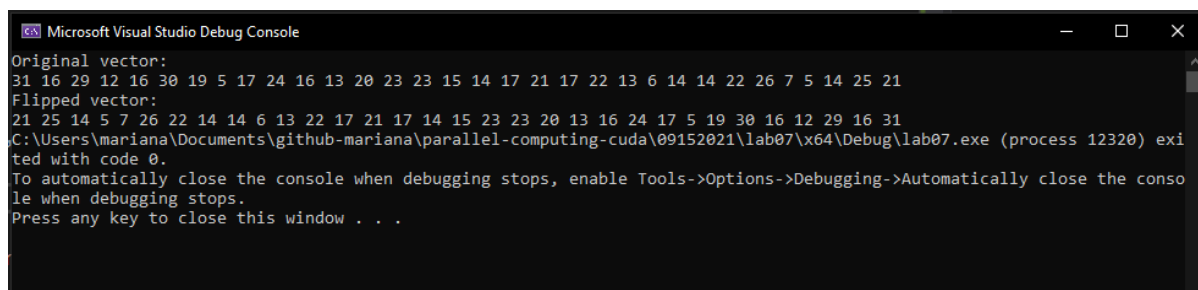
- 32 threads
- 1 block of 1 dimension
- The kernel must be: `__global__ void flipVector(int* vector, int* flippedVector)`
- Include error management using the following function: `__host__ void checkCUDAError(const char* msg)`

Solution

```
1  #include "cuda_runtime.h"
2  #include "device_launch_parameters.h"
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  #include <stdlib.h> /* srand, rand */
8  #include <time.h> /* time */
9
10 __host__ void checkCUDAError(const char* msg) {
11     cudaError_t error;
12     cudaDeviceSynchronize();
13     error = cudaGetLastError();
14     if (error != cudaSuccess) {
15         printf("ERROR %d: %s (%s)\n", error,
16             cudaGetErrorString(error), msg);
17     }
18 }
19
20 __global__ void flipVector(int* vector, int* flippedVector) {
21     int gId = threadIdx.x + blockIdx.x * blockDim.x;
22     flippedVector[(blockDim.x - 1) - gId] = vector[gId];
23 }
```

```
24 int main() {
25     const int vectorSize = 32;
26     int* vector = (int*)malloc(sizeof(int) * vectorSize);
27     int* flippedVector = (int*)malloc(sizeof(int) * vectorSize
    );
28
29     int* devVector, * devFlippedVector;
30     cudaMalloc((void**)&devVector, sizeof(int) * vectorSize);
31     checkCUDAError("cudaMalloc: devVector");
32     cudaMalloc((void**)&devFlippedVector, sizeof(int) *
    vectorSize);
33     checkCUDAError("cudaMalloc: devFlippedVector");
34
35     srand(time(NULL));
36     printf("Original vector: \n");
37     for (int i = 0; i < vectorSize; i++) {
38         int num = rand() % vectorSize + 1;
39         vector[i] = num;
40         printf("%d ", vector[i]);
41     }
42
43     cudaMemcpy(flippedVector, vector, sizeof(int) * vectorSize
    , cudaMemcpyHostToHost);
44     checkCUDAError("cudaMemcpy: vector -> flippedVector, Host
    -> Host");
45     cudaMemcpy(devVector, vector, sizeof(int) * vectorSize,
    cudaMemcpyHostToDevice);
46     checkCUDAError("cudaMemcpy: vector -> devVector, Host ->
    Device");
47     cudaMemcpy(devFlippedVector, flippedVector, sizeof(int) *
    vectorSize, cudaMemcpyHostToDevice);
48     checkCUDAError("cudaMemcpy: flippedVector ->
    devFlippedVector, Host -> Device");
49
50     dim3 grid(1);
51     dim3 block(vectorSize);
52
53     flipVector << < grid, block >> > (devVector,
    devFlippedVector);
54     checkCUDAError("kernel: flipVector");
55
56     cudaMemcpy(flippedVector, devFlippedVector, sizeof(int) *
    vectorSize, cudaMemcpyDeviceToHost);
57     checkCUDAError("cudaMemcpy: devFlippedVector ->
    flippedVector, Device -> Host");
58
59     printf("\nFlipped vector: \n");
60     for (int i = 0; i < vectorSize; i++) {
61         printf("%d ", flippedVector[i]);
62     }
63 }
```

Output



```
Microsoft Visual Studio Debug Console
Original vector:
31 16 29 12 16 30 19 5 17 24 16 13 20 23 23 15 14 17 21 17 22 13 6 14 14 22 26 7 5 14 25 21
Flipped vector:
21 25 14 5 7 26 22 14 14 6 13 22 17 21 17 14 15 23 23 20 13 16 24 17 5 19 30 16 12 29 16 31
C:\Users\mariana\Documents\github-mariana\parallel-computing-cuda\09152021\lab07\x64\Debug\lab07.exe (process 12320) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Figure 1: Image