

Universidad Nacional de San Agustín de Arequipa

AÑO DE LA RECUPERACIÓN Y CONSOLIDACIÓN DE LA  
ECONOMÍA PERUANA



ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN  
Temas en Inteligencia Artificial

---

## LABORATORIO 1

PROGRAMACIÓN EN CUDA

A-2025

---

Profesor: MSc. R. Jesús Cárdenas Talavera

Javier Wilber Quispe Rojas

# Índice

Índice	1
1 Implementar el Hola Mundo desde un kernel de CUDA	2
2 Elaborar un programa para CPU que sume dos vectores de valores flotantes. Mida el tiempo de ejecución para sumar vectores mayores a 1000 elementos.	2
3 Elaborar un programa que utilice la GPU para la suma de dos vectores de valores flotantes. Mida el tiempo de ejecución para sumar vectores mayores a 1000 elementos.	4

## 1. Implementar el Hola Mundo desde un kernel de CUDA

```
1 #include<stdio.h>
2 __global__ void hello(void)
3 {
4     printf("GPU: -Hola-Mundo!\n");
5 }
6 int main(int argc, char **argv)
7 {
8
9     hello<<<1,10>>>>();
10    cudaDeviceReset();
11    return 0;
12 }
```

Listing 1: Código ejercicio 1

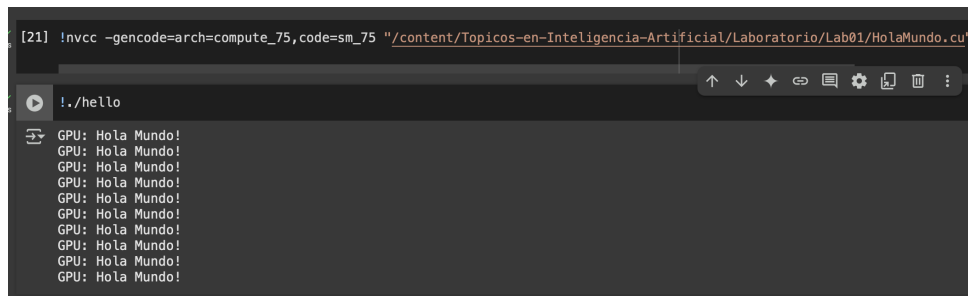


Figura 1: Ejecución

## 2. Elaborar un programa para CPU que sume dos vectores de valores flotantes. Mida el tiempo de ejecución para sumar vectores mayores a 1000 elementos.

```
1 #include "iostream"
2 #include <cstdlib>
3 #include <vector>
4 #include <math.h>
5 #include <chrono>
6
7 using namespace std;
8 const int N=1000;
9 std::vector<float> A(N), B(N), C(N);
10 void CrearVectores(vector<float>& A, vector<float>& B){
11     srand(time(NULL));
12     for(int i=0; i<N; i++){
13         A[i] = (rand() % 1000)/10.0;
14         B[i] = (rand() % 1000)/10.0;
15     }
16 }
17
18 float Tiempo(vector<float>& A, vector<float>& B, vector<float>& C){
19     float time=0.0;
20     auto start = std::chrono::high_resolution_clock::now();
21
22     for (int i = 0; i < N; ++i) {
23         C[i] = A[i] + B[i];
24     }
25
26     auto end = std::chrono::high_resolution_clock::now();
27     std::chrono::duration<float, std::milli> duration = end - start;
28
29     time=duration.count();
30
31     return time;
32 }
```

```

33 }
34 }
35
36 int main() {
37     CrearVectores(A,B);
38
39     float time =Tiempo(A,B,C);
40
41
42     for(int i=0; i<1000 ; i++)
43     {
44         cout<<"A"<<i<<" : - "<<A[i]<<" + - "<<"B"<<i<<" : - "<<B[i]<<" = - "<<"C"<<i<<" : - "<<C[i]<<
45         endl;
46     }
47     cout<<"El tiempo en segundos fue : - "<<time<<endl;
48     return 0;
49 }

```

Listing 2: Codigo ejercicio 2

```

[24] !g++ "/content/Topicos-en-Inteligencia-Artificial/Laboratorio/Lab01/E2.cpp" -o E2
!./E2
CPU A0: 64.2 + B0: 81.7 = C0: 145.9
CPU A1: 38 + B1: 9.6 = C1: 47.6
CPU A2: 52.8 + B2: 93.2 = C2: 146
CPU A3: 41.6 + B3: 27.8 = C3: 69.4
CPU A4: 58.7 + B4: 8.4 = C4: 67.1
CPU A5: 78.1 + B5: 37.8 = C5: 115.9
CPU A6: 38.7 + B6: 5.4 = C6: 44.1
CPU A7: 80.3 + B7: 42 = C7: 122.3
CPU A8: 34.3 + B8: 67.5 = C8: 101.8
CPU A9: 74 + B9: 13.9 = C9: 87.9
CPU A10: 40 + B10: 54.8 = C10: 94.8
CPU A11: 80.4 + B11: 60.7 = C11: 141.1
CPU A12: 79 + B12: 60.6 = C12: 139.6
CPU A13: 92.8 + B13: 53.8 = C13: 146.6
CPU A14: 44.5 + B14: 13.8 = C14: 58.3
CPU A15: 36.1 + B15: 8.8 = C15: 44.9
CPU A16: 30.7 + B16: 9.3 = C16: 40
CPU A17: 53.6 + B17: 18.8 = C17: 72.4
CPU A18: 2.6 + B18: 95.2 = C18: 97.8
CPU A19: 46.6 + B19: 96.5 = C19: 143.1
CPU A20: 38.8 + B20: 59.9 = C20: 98.7

```

Figura 2: Ejecución ejercicio 2

```

CPU A977: 81.3 + B977: 80.6 = C977: 161.9
CPU A978: 65.5 + B978: 44.1 = C978: 109.6
CPU A979: 12.1 + B979: 31.5 = C979: 43.6
CPU A980: 51 + B980: 65.1 = C980: 116.1
CPU A981: 5.8 + B981: 72.9 = C981: 78.7
CPU A982: 18.9 + B982: 28.8 = C982: 47.7
CPU A983: 49.2 + B983: 50.6 = C983: 99.8
CPU A984: 19.9 + B984: 8.8 = C984: 28.7
CPU A985: 66.8 + B985: 4.5 = C985: 71.3
CPU A986: 55.9 + B986: 8.3 = C986: 64.2
CPU A987: 20.6 + B987: 37.6 = C987: 58.2
CPU A988: 15.5 + B988: 79.5 = C988: 95
CPU A989: 79.4 + B989: 36.3 = C989: 115.7
CPU A990: 14.1 + B990: 69.1 = C990: 83.2
CPU A991: 84.6 + B991: 44.6 = C991: 129.2
CPU A992: 72.2 + B992: 1.2 = C992: 73.4
CPU A993: 25.2 + B993: 72.9 = C993: 98.1
CPU A994: 45.3 + B994: 72.5 = C994: 117.8
CPU A995: 39.6 + B995: 31.5 = C995: 71.1
CPU A996: 37.6 + B996: 45.5 = C996: 83.1
CPU A997: 4.5 + B997: 91.8 = C997: 96.3
CPU A998: 9.5 + B998: 53.7 = C998: 63.2
CPU A999: 42.4 + B999: 29.4 = C999: 71.8
El tiempo en segundos fue: 0.011092

```

Figura 3: Ejecución ejercicio 2

### 3. Elaborar un programa que utilice la GPU para la suma de dos vectores de valores flotantes. Mida el tiempo de ejecución para sumar vectores mayores a 1000 elementos.

```
1  #include <iostream>
2  #include <vector>
3  #include <cstdlib>
4  #include <cuda_runtime.h>
5  #include <chrono>
6
7  using namespace std;
8
9  const int N = 1000;
10 vector<float> A(N), B(N), C(N);
11
12
13 __global__ void vectorAddGPU(const float* A, const float* B, float* C, int N) {
14     int i = blockIdx.x * blockDim.x + threadIdx.x;
15     if (i < N)
16         C[i] = A[i] + B[i];
17 }
18
19
20 void CrearVectores(vector<float>& A, vector<float>& B) {
21     srand(time(NULL));
22     for (int i = 0; i < N; i++) {
23         A[i] = (rand() % 1000) / 10.0f;
24         B[i] = (rand() % 1000) / 10.0f;
25     }
26 }
27
28
29 float TiempoGPU(vector<float>& A, vector<float>& B, vector<float>& C) {
30     float *d_A, *d_B, *d_C;
31     size_t size = N * sizeof(float);
32
33     cudaMalloc(&d_A, size);
34     cudaMalloc(&d_B, size);
35     cudaMalloc(&d_C, size);
36
37     cudaMemcpy(d_A, A.data(), size, cudaMemcpyHostToDevice);
38     cudaMemcpy(d_B, B.data(), size, cudaMemcpyHostToDevice);
39
40     int threadsPerBlock = 256;
41     int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;
42
43     auto start = std::chrono::high_resolution_clock::now();
44
45     vectorAddGPU<<<blocksPerGrid, threadsPerBlock>>>>(d_A, d_B, d_C, N);
46     cudaDeviceSynchronize();
47
48     auto end = std::chrono::high_resolution_clock::now();
49     std::chrono::duration<float, std::milli> duration = end - start;
50
51     cudaMemcpy(C.data(), d_C, size, cudaMemcpyDeviceToHost);
52
53     cudaFree(d_A);
54     cudaFree(d_B);
55     cudaFree(d_C);
56
57     return duration.count();
58 }
59
60 int main() {
61     CrearVectores(A, B);
62
63     float time = TiempoGPU(A, B, C);
64
65     for (int i = 0; i < 1000; i++) {
66         cout << "GPU-" << "A" << i << " : -" << A[i]
67              << " -+B" << i << " : -" << B[i]
```

```

68         << "-C" << i << ":-" << C[i] << endl;
69     }
70
71     cout << "El tiempo de ejecuci n en GPU fue:-" << time << "-ms" << endl;
72
73     return 0;
74 }

```

Listing 3: Código ejercicio 3

```

[18] !nvcc -gencode=arch=compute_75,code=sm_75 "/content/Topicos-en-Inteligencia-Artificial/Laboratorio/Lab01/E3.cu" -o e3

!./e3

A0: 76.7 + B0: 47.1 = C0: 123.8
A1: 6.6 + B1: 29.8 = C1: 36.4
A2: 51.5 + B2: 43 = C2: 94.5
A3: 93.9 + B3: 64.4 = C3: 158.3
A4: 25.5 + B4: 50.8 = C4: 76.3
A5: 39.2 + B5: 49.8 = C5: 89
A6: 43.8 + B6: 42.8 = C6: 86.6
A7: 40.5 + B7: 51.1 = C7: 91.6
A8: 65.8 + B8: 76.2 = C8: 142
A9: 59.8 + B9: 11 = C9: 70.8
A10: 0.8 + B10: 72 = C10: 72.8
A11: 51 + B11: 12.7 = C11: 63.7
A12: 77.7 + B12: 16.9 = C12: 94.6
A13: 3.1 + B13: 88.2 = C13: 91.3
A14: 25.3 + B14: 29.6 = C14: 54.9
A15: 70.3 + B15: 37.2 = C15: 107.5
A16: 76.8 + B16: 76.9 = C16: 153.7
A17: 2.2 + B17: 63.5 = C17: 65.7
A18: 55.1 + B18: 31.3 = C18: 86.4
A19: 28 + B19: 15.9 = C19: 43.9
A20: 82.2 + B20: 2.4 = C20: 84.6

```

Figura 4: Ejecución ejercicio 3

```

A980: 69.7 + B980: 49.5 = C980: 119.2
A981: 78.9 + B981: 44.3 = C981: 123.2
A982: 55.3 + B982: 11.5 = C982: 66.8
A983: 10.5 + B983: 30.8 = C983: 41.3
A984: 28.9 + B984: 73.6 = C984: 102.5
A985: 7.2 + B985: 64.9 = C985: 72.1
A986: 95.4 + B986: 89.5 = C986: 184.9
A987: 84.5 + B987: 0.4 = C987: 84.9
A988: 67.6 + B988: 88.2 = C988: 155.8
A989: 96.4 + B989: 37.8 = C989: 134.2
A990: 32.5 + B990: 10.7 = C990: 43.2
A991: 72.9 + B991: 99.9 = C991: 172.8
A992: 73.6 + B992: 53.2 = C992: 126.8
A993: 63 + B993: 52.5 = C993: 115.5
A994: 10.8 + B994: 63.4 = C994: 74.2
A995: 82.2 + B995: 15.7 = C995: 97.9
A996: 13 + B996: 61.1 = C996: 74.1
A997: 60.1 + B997: 3.5 = C997: 63.6
A998: 7.8 + B998: 5.8 = C998: 13.6
A999: 34.4 + B999: 72 = C999: 106.4
El tiempo de ejecución en GPU fue: 0.200453 ms

```

Figura 5: Ejecución ejercicio 3