



National University
Of Computer and Emerging Sciences

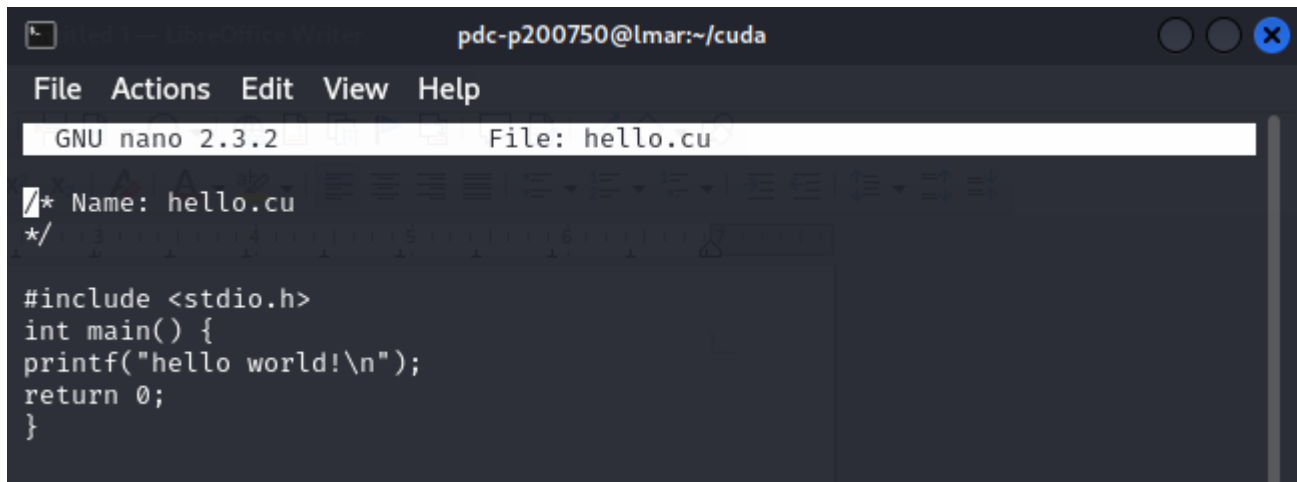
Name : Waqar Ahmed

Roll No: 20P-0750

Assignment #02

Task 1 : Login successfully

Task 2: Hello World



A screenshot of a terminal window with a nano text editor. The window title is "pdc-p200750@lmar:~/cuda". The nano editor shows the file "hello.cu" with the following C code:

```
#include <stdio.h>
int main() {
printf("hello world!\n");
return 0;
}
```

```
pdc-p200750@lmar ~ $ scp hello.cu pdc-p200750@121.52.146.108:/home/pdc-p200750/cuda
(pdc-p200750@121.52.146.108) Password:
hello.cu                               100%  92  467.4KB/s   00:00
pdc-p200750@lmar ~ $ cd cuda
pdc-p200750@lmar ~/cuda $ ls -lh
total 4.0K
-rw-r--r-- 1 pdc-p200750 pdc-p200750 92 Dec 19 21:02 hello.cu
pdc-p200750@lmar ~/cuda $ nvcc hello.cu
pdc-p200750@lmar ~/cuda $ ./a.out
hello world!
pdc-p200750@lmar ~/cuda $
```

Task 5: Playing with 1D GPU indices

1. Task 5a

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cdc-p200750@lmar ~/cuda $
```

2. Task 5b

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
cdc-p200750@lmar ~/cuda $
```

3. Task 5c

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
cdc-p200750@lmar ~/cuda $
```

4. Task 5d

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
cdc-p200750@lmar ~/cuda $
```

5. Task 5e

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0
pdc-p200750@lmar ~/cuda $
```

6. Task 5f

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 0 1 2 3 4 5 6 7
pdc-p200750@lmar ~/cuda $
```

7. Task 5g

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888
pdc-p200750@lmar ~/cuda $
```

8. Task 5h: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0
pdc-p200750@lmar ~/cuda $
```

Position 2: myHelloOnGPU<<<N/2, 1>>>(gpuArray);

Position 1: array[blockIdx.x * 2] = 111 *(blockIdx.x + 1);

9. Task 5j

```
pdc-p200750@lmar ~/cuda $ nvcc task5.cu
pdc-p200750@lmar ~/cuda $ ./a.out
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
pdc-p200750@lmar ~/cuda $
```

10. Task 5k

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0
cdc-p200750@lmar ~/cuda $
```

11. Task 5m

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444
cdc-p200750@lmar ~/cuda $
```

12. Task 5n: What should be Position 1 and 2 in order to obtain the following output:

```
cdc-p200750@lmar ~/cuda $ nvcc task5.cu
cdc-p200750@lmar ~/cuda $ ./a.out
3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
cdc-p200750@lmar ~/cuda $
```

Position 2: myHelloOnGPU<<<N/4, N/4>>>(gpuArray);

Position 1: array[blockIdx.x * blockDim.x + threadIdx.x] = blockDim.x – threadIdx.x – 1;

Task 6: Playing with 2D GPU indices.

1. Task 6a:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
```

2. Task 6b:

```
pdc-p200750@lmar ~/cuda $ nano task6.cu
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
```

3. Task 6c:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
00 01 02 03
04 05 06 07
08 09 10 11
12 13 14 15
```

4. Task 6d:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 22 33 44
00 00 00 00
00 00 00 00
00 00 00 00
```

5. Task 6e:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00
```

6. Task 6f:

```
pdc-p200750@lmar ~/cuda $ ./a.out
11 00 00 00
00 22 00 00
00 00 33 00
00 00 00 44
```

for above output we need to change in position 1 and position 2 as following.

Position1 : `int index = threadIdx.x * blockDim.x;`

`array[index] = (index % 5 == 0) ? (index / 5 + 1) * 11 : 0;`

Position2: `dim3 dimGrid(4,1,1); dim3 blockDim(4,1,1);`

7. Task 6g:

```
pdg-p200750@lmar ~/cuda $ nvcc task6.cu
pdg-p200750@lmar ~/cuda $ ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44
```

8. Task 6f:

```
pdg-p200750@lmar ~/cuda $ nvcc task6.cu
pdg-p200750@lmar ~/cuda $ ./a.out
11 00 00 00
22 00 00 00
33 00 00 00
44 00 00 00
```

9. Task 6g:

```
pdg-p200750@lmar ~/cuda $ nvcc task6.cu
pdg-p200750@lmar ~/cuda $ ./a.out
00 00 00 00
00 00 00 00
00 00 00 00
11 22 33 44
```

10. Task 6h:

```
pdg-p200750@lmar ~/cuda $ nvcc task6.cu
pdg-p200750@lmar ~/cuda $ ./a.out
00 00 00 11
00 00 00 22
00 00 00 33
00 00 00 44
```


11. Task 6j:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 22 33 44
11 22 33 44
11 22 33 44
11 22 33 44
```

12. Task 6k:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
11 11 11 11
22 22 22 22
33 33 33 33
44 44 44 44
```

13. Task 6m: What should be Position 1 and 2 in order to obtain the following output:

```
pdc-p200750@lmar ~/cuda $ nvcc task6.cu
pdc-p200750@lmar ~/cuda $ ./a.out
44 44 44 44
33 33 33 33
22 22 22 22
11 11 11 11
```

Position2: `dim3 dimGrid(N/4,1,1) ; dim3 dimBlock(N/4 , 1 , 1);`

Position1: `int index = threadIdx.x + blockIdx.x * blockDim.x;`

`array[index] = (4- blockIdx.x) * 11;`

14. Task 6n:

```
cdc-p200750@lmar ~/cuda $ nvcc task6.cu
cdc-p200750@lmar ~/cuda $ ./a.out
11 11 22 22
11 11 22 22
33 33 44 44
33 33 44 44
```

15. Task 6o: What should be Position 1 and 2 in order to obtain the following output:

Task 7: Matrix Addition

```

pdc-p200750@lmar:~/cuda
File Actions Edit View Help
GNU nano 2.3.2 File: matrix-add.cu

#include <stdio.h>
#include <stdlib.h>

__global__ void add(int *a, int *b, int *c) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;

    // Perform matrix addition
    c[idx] = a[idx] + b[idx];
}

int main() {
    int *a, *b, *c, *da, *db, *dc, N = 16, i;

    a = (int*)malloc(sizeof(int) * N);
    b = (int*)malloc(sizeof(int) * N);
    c = (int*)malloc(sizeof(int) * N);

    // Initialize matrices a and b to 1's
    for (i = 0; i < N; i++) {
        a[i] = 1;
        b[i] = 1;
    }
}

```

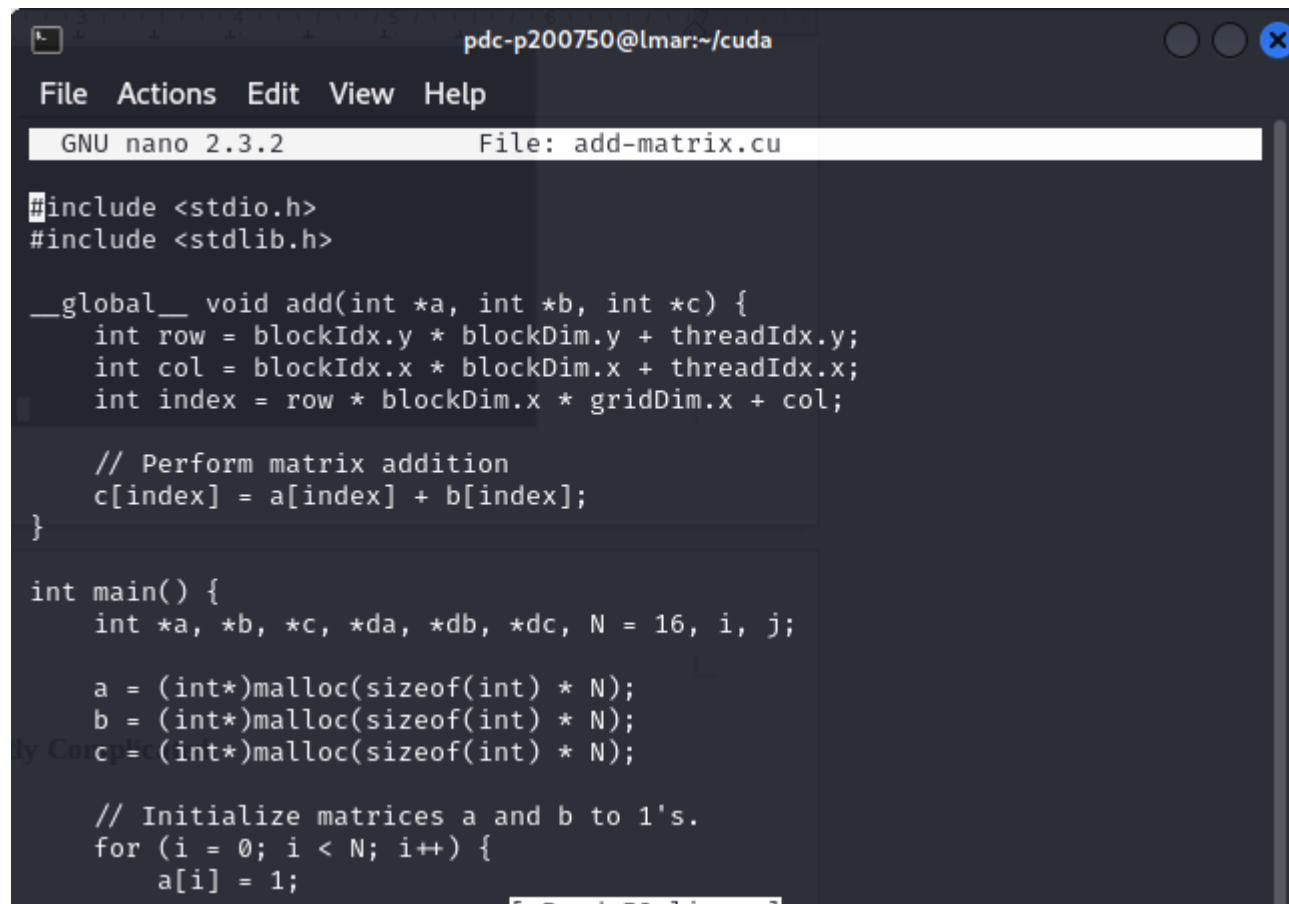
output:

```

pdc-p200750@lmar ~/cuda $ nvcc matrix-add.cu
pdc-p200750@lmar ~/cuda $ ./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] = 0
a[9] + b[9] = 0
a[10] + b[10] = 0
a[11] + b[11] = 0
a[12] + b[12] = 0
a[13] + b[13] = 0
a[14] + b[14] = 0
a[15] + b[15] = 0
pdc-p200750@lmar ~/cuda $

```

Task 8: Matrix Addition Slightly Complicated



The screenshot shows a terminal window with a dark background. The title bar at the top reads "pdc-p200750@lmar:~/cuda". Below the title bar is a menu bar with "File", "Actions", "Edit", "View", and "Help". A status bar at the top of the editor area shows "GNU nano 2.3.2" and "File: add-matrix.cu". The code is written in C++ and CUDA. It includes `<stdio.h>` and `<stdlib.h>`. A global function `add` is defined, which takes three integer arrays `a`, `b`, and `c` as arguments. Inside the function, the row and column indices for the current thread are calculated using `blockIdx`, `blockDim`, and `threadIdx`. The index of the element to be added is calculated as `row * blockDim.x * gridDim.x + col`. The element at this index in array `c` is then set to the sum of the corresponding elements in arrays `a` and `b`. The `main` function declares three integer arrays `a`, `b`, and `c` of size `N = 16`. It then allocates memory for each array using `malloc`. Finally, it initializes array `a` with the value 1 for all elements.

```
pdc-p200750@lmar:~/cuda
File Actions Edit View Help
GNU nano 2.3.2 File: add-matrix.cu

#include <stdio.h>
#include <stdlib.h>

__global__ void add(int *a, int *b, int *c) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    int index = row * blockDim.x * gridDim.x + col;

    // Perform matrix addition
    c[index] = a[index] + b[index];
}

int main() {
    int *a, *b, *c, *da, *db, *dc, N = 16, i, j;

    a = (int*)malloc(sizeof(int) * N);
    b = (int*)malloc(sizeof(int) * N);
    c = (int*)malloc(sizeof(int) * N);

    // Initialize matrices a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = 1;
    }
}
```

```
GNU nano 2.3.2      File: add-matrix.cu

int main() {
    int *a, *b, *c, *da, *db, *dc, N = 16, i, j;

    a = (int*)malloc(sizeof(int) * N);
    b = (int*)malloc(sizeof(int) * N);
    c = (int*)malloc(sizeof(int) * N);

    // Initialize matrices a and b to 1's.
    for (i = 0; i < N; i++) {
        a[i] = 1;
        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, N/8, 1);
    dim3 dimBlock(N/8, N/8, 1);
}
```

output:

```
cdc-p200750@lmar ~/cuda $ nvcc add-matrix.cu
cdc-p200750@lmar ~/cuda $ ./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

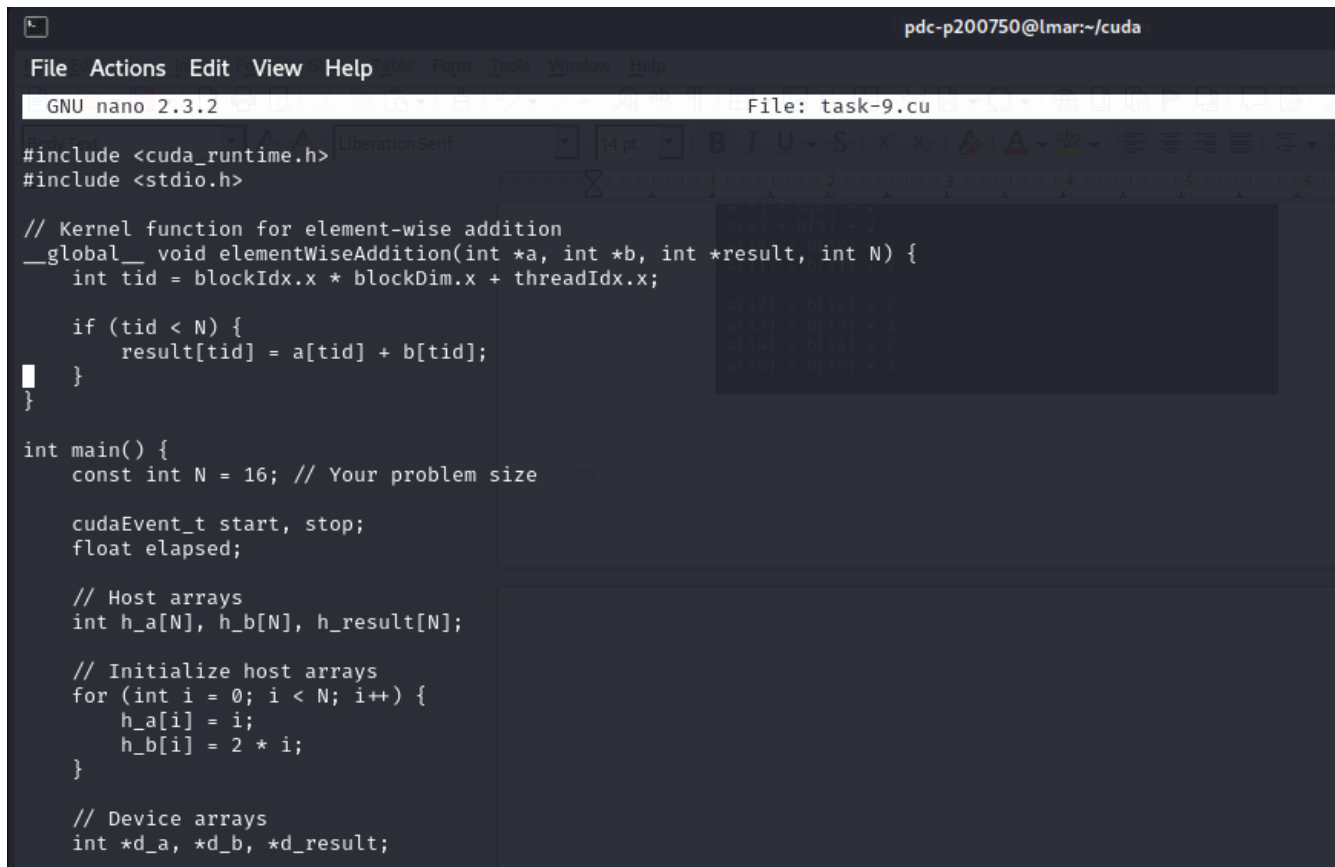
a[8] + b[8] = 2
a[9] + b[9] = 2
a[10] + b[10] = 2
a[11] + b[11] = 2

a[12] + b[12] = 2
a[13] + b[13] = 2
a[14] + b[14] = 2
a[15] + b[15] = 2
```

Task 9: Measurements

To measure anything in CUDA, we can use the following from the CUDA Events API's:

code:



The screenshot shows a terminal window with the nano text editor. The title bar indicates the user is 'pdc-p200750@lmar' in the directory '~/cuda'. The editor is running GNU nano 2.3.2 and is editing a file named 'task-9.cu'. The code is a CUDA program for element-wise addition of two arrays. It includes `<cuda_runtime.h>` and `<stdio.h>`. A kernel function `elementWiseAddition` is defined, which takes pointers to integer arrays `a`, `b`, and `result`, and the size `N`. It uses `blockIdx.x` and `threadIdx.x` to calculate the thread ID `tid` and performs the addition `result[tid] = a[tid] + b[tid]` if `tid` is less than `N`. The `main` function sets `N = 16`, declares host arrays `h_a`, `h_b`, and `h_result`, initializes `h_a` with values 0 to 15 and `h_b` with values 0 to 31, and declares device arrays `d_a`, `d_b`, and `d_result`. A CUDA event timer is set up to measure the execution time of the kernel. The code is as follows:

```
#include <cuda_runtime.h>
#include <stdio.h>

// Kernel function for element-wise addition
__global__ void elementWiseAddition(int *a, int *b, int *result, int N) {
    int tid = blockIdx.x * blockDim.x + threadIdx.x;

    if (tid < N) {
        result[tid] = a[tid] + b[tid];
    }
}

int main() {
    const int N = 16; // Your problem size

    cudaEvent_t start, stop;
    float elapsed;

    // Host arrays
    int h_a[N], h_b[N], h_result[N];

    // Initialize host arrays
    for (int i = 0; i < N; i++) {
        h_a[i] = i;
        h_b[i] = 2 * i;
    }

    // Device arrays
    int *d_a, *d_b, *d_result;
```

```
File Actions Edit View Help
GNU nano 2.3.2 File: task-9.cu
cudaMalloc((void **) &d_a, N * sizeof(int));
cudaMalloc((void **) &d_b, N * sizeof(int));
cudaMalloc((void **) &d_result, N * sizeof(int));

// Copy data from host to device
cudaMemcpy(d_a, h_a, N * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_b, h_b, N * sizeof(int), cudaMemcpyHostToDevice);

printf("Grid X\tGrid Y\tGrid Z\tBlock X\tBlock Y\tBlock Z\tMicroseconds\tMilliseconds\tSeconds\n");

for (int gridX = 1; gridX ≤ N; gridX *= 2) {
    for (int gridY = 1; gridY ≤ N / gridX; gridY *= 2) {
        int gridZ = N / (gridX * gridY);

        for (int blockX = 1; blockX ≤ N; blockX *= 2) {
            for (int blockY = 1; blockY ≤ N / blockX; blockY *= 2) {
                int blockZ = N / (blockX * blockY);

                dim3 gridSize(gridX, gridY, gridZ);
                dim3 blockSize(blockX, blockY, blockZ);

                cudaEventCreate(&start);
                cudaEventCreate(&stop);

                cudaEventRecord(start, 0);

                // Call the Kernel
                elementWiseAddition<<<gridSize, blockSize>>>(d_a, d_b, d_result, N);
            }
        }
    }
}
```

output:

```
pd-c-p200750@lmar ~/cuda $ nvcc task-9.cu
pd-c-p200750@lmar ~/cuda $ ./a.out
Grid X Grid Y Grid Z Block X Block Y Block Z Microseconds Milliseconds Seconds
1 1 16 1 1 16 23.94 0.02 0.000024
1 1 16 1 2 8 8.70 0.01 0.000009
1 1 16 1 4 4 6.08 0.01 0.000006
1 1 16 1 8 2 5.79 0.01 0.000006
1 1 16 1 16 1 5.79 0.01 0.000006
1 1 16 2 1 8 5.79 0.01 0.000006
1 1 16 2 2 4 5.95 0.01 0.000006
1 1 16 2 4 2 5.92 0.01 0.000006
1 1 16 2 8 1 5.76 0.01 0.000006
1 1 16 4 1 4 5.76 0.01 0.000006
1 1 16 4 2 2 5.73 0.01 0.000006
1 1 16 4 4 1 5.76 0.01 0.000006
1 1 16 8 1 2 5.98 0.01 0.000006
1 1 16 8 2 1 5.73 0.01 0.000006
1 1 16 16 1 1 5.76 0.01 0.000006
1 2 8 1 1 16 5.79 0.01 0.000006
1 2 8 1 2 8 5.76 0.01 0.000006
1 2 8 1 4 4 5.73 0.01 0.000006
1 2 8 1 8 2 5.79 0.01 0.000006
1 2 8 1 16 1 5.73 0.01 0.000006
1 2 8 2 1 8 5.76 0.01 0.000006
1 2 8 2 2 4 5.95 0.01 0.000006
1 2 8 2 4 2 5.76 0.01 0.000006
1 2 8 2 8 1 5.76 0.01 0.000006
1 2 8 4 1 4 5.79 0.01 0.000006
1 2 8 4 2 2 5.76 0.01 0.000006
1 2 8 4 4 1 5.73 0.01 0.000006
1 2 8 8 1 2 5.86 0.01 0.000006
Pages 13 and 14 of 14 274 words, 1,525 characters Default Page Style English (USA)
```