```
1 %load_ext nvcc_plugin
created output directory at /content/src
Out bin /content/result.out
```

Perform cuda operations with 1 thread and 1 block

```
[ ] 4 1 cell hidden
```

Perform CUDA operations on blockldx

```
%%CU
    #include <stdio.h>
    #define N 512
 4
 5
     __global__ void
    add(int *a, int *b, int *c) {
 6
7
        c[blockIdx.x] = a[blockIdx.x] + b[blockIdx.x];
8
    }
9
10
    int main(void) {
        int *a, *b, *c; // The arrays on the host CPU machine
11
        int *d_a, *d_b, *d_c; // The arrays for the GPU device
12
13
14
        int size = N * sizeof(int);
15
16
        a = (int *)malloc(size);
17
        b = (int *)malloc(size);
        c = (int *)malloc(size);
18
19
20
        for (int i = 0; i < N; i++) {
21
             a[i] = i;
22
             b[i] = i;
23
        }
24
25
        cudaMalloc((void **)&d a, size);
26
        cudaMalloc((void **)&d b, size);
        cudaMalloc((void **)&d_c, size);
27
28
29
        cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
30
        cudaMemcpy(d b, b, size, cudaMemcpyHostToDevice);
31
32
        add<<<N, 1>>>(d_a, d_b, d_c);
33
34
        cudaMemcpy(c, d c, size, cudaMemcpyDeviceToHost);
35
36
        for (int i = 0; i < N; i++) {
           if (c[i] \mid = a[i] + h[i]) {
37
```

```
✓ 1s
                                 completed at 3:05 PM
39
           }
40
         }
41
42
         free(a);
43
         free(b);
44
         free(c);
45
46
         cudaFree(d a);
47
         cudaFree(d_b);
48
         cudaFree(d c);
49
50
         return 0;
51
     }
```

Perform CUDA operations on threadIdx

```
1 %%cu
 2 #include <stdio.h>
 3 #define N 512
 5 __global__ void
 6 add(int *a, int *b, int *c) {
 7
      c[threadIdx.x] = a[threadIdx.x] + b[threadIdx.x];
 8 }
9
10 int main(void) {
      int *a, *b, *c;
                              // The arrays on the host CPU machine
11
      int *d a, *d b, *d c; // The arrays for the GPU device
12
13
14
      int size = N * sizeof(int);
15
16
      a = (int *)malloc(size);
17
      b = (int *)malloc(size);
18
      c = (int *)malloc(size);
19
20
      for (int i = 0; i < N; i++) {
21
           a[i] = i;
           b[i] = i;
22
23
      }
24
25
      cudaMalloc((void **)&d a, size);
26
      cudaMalloc((void **)&d b, size);
27
      cudaMalloc((void **)&d c, size);
28
29
      cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
30
      cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);
31
32
      add<<<1, N>>>(d_a, d_b, d_c);
33
      cudaMamony/c d c ciza cudaMamonyDavicaToHoct).
```

```
cauariemepy(c, a_c, size, cauariemepypeviceronose),
35
36
       for (int i = 0; i < N; i++) {
         if (c[i] != a[i] + b[i]) {
37
             printf("Error in calculation;");
38
39
         }
       }
40
41
42
       free(a);
       free(b);
43
44
       free(c);
45
46
      cudaFree(d a);
47
      cudaFree(d b);
       cudaFree(d c);
48
49
50
       return 0;
51 }
```

Perform CUDA operations with thread and blocks combined

```
1
    %%CU
   #include <stdio.h>
   #define N (2048*2048)
    #define TPB 512
 5
6
    __global__ void
7
    add(int *a, int *b, int *c) {
8
        int index = threadIdx.x + blockIdx.x * blockDim.x;
9
        c[index] = a[index] + b[index];
10
    }
11
12
    int main(void) {
        int *a, *b, *c;
                            // The arrays on the host CPU machine
13
        int *d a, *d b, *d c; // The arrays for the GPU device
14
15
16
        int size = N * sizeof(int);
17
18
        a = (int *)malloc(size);
19
        b = (int *)malloc(size);
20
        c = (int *)malloc(size);
21
22
        for (int i = 0; i < N; i++) {
23
            a[i] = i;
24
            b[i] = i;
25
        }
26
```

```
cuuamactuc((vuiu "")au_a, Size);
۷1
28
        cudaMalloc((void **)&d_b, size);
29
        cudaMalloc((void **)&d_c, size);
30
31
        cudaMemcpy(d a, a, size, cudaMemcpyHostToDevice);
        cudaMemcpy(d b, b, size, cudaMemcpyHostToDevice);
32
33
34
        add << N/TPB, TPB>>> (d a, d b, d c);
35
36
        cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);
37
        for (int i = 0; i < N; i++) {
38
39
          if (c[i] != a[i] + b[i]) {
               printf("Error in calculation;");
40
41
          }
        }
42
43
44
        free(a);
45
        free(b);
46
        free(c);
47
        cudaFree(d a);
48
49
        cudaFree(d_b);
        cudaFree(d_c);
50
51
52
        return 0;
53
    }
```

1

Colab paid products - Cancel contracts here