```
!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

```
code = """
#include <stdio.h>
#include <stdlib.h>

#define N 1024
#define BLOCK_SIZE 16

__global__ void matrixMul(int *a, int *b, int *c, int width) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    int sum = 0;
    for (int i = 0; i < width; i++) {
        sum += a[row * width + i] * b[i * width + col];
    }
    c[row * width + col] = sum;
}

int main() {
    int *a, *b, *c;
    int *d_a, *d_b, *d_c;
    int size = N * N * sizeof(int);

    // Allocate memory on host
    a = (int*)malloc(size);
    b = (int*)malloc(size);
    c = (int*)malloc(size);

    // Initialize matrices
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            a[i * N + j] = i + j;
            b[i * N + j] = i - j;
        }
    }

    // Allocate memory on device
    cudaMalloc(&d_a, size);
    cudaMalloc(&d_b, size);
    cudaMalloc(&d_c, size);

    // Copy data from host to device
    cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);

    // Launch kernel with 2D grid and 2D block
    dim3 dimBlock(BLOCK_SIZE, BLOCK_SIZE);
    dim3 dimGrid((N + dimBlock.x - 1) / dimBlock.x, (N + dimBlock.y - 1) / dimBlock.y);
    matrixMul<<<dimGrid, dimBlock>>>(d_a, d_b, d_c, N);

    // Copy result from device to host
    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    // Print first and last elements of result
    printf("c[0][0] = %d, c[%d][%d] = %d", c[0], N-1, N-1, c[(N-1) * N + (N-1)]);

    // Free memory
    free(a);
    free(b);
    free(c);
    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}


"""


text_file = open("assign4b.cu","w")
text_file.write(code)
text_file.close()
```

```
!nvcc assign4b.cu
```

```
!./a.out
```

```
c[0][0] = 357389824, c[1023][1023] = -714255872
```

```
!nvprof ./a.out
```

```
!nvprof ./a.out
```

```
==939== NVPROF is profiling process 939, command: ./a.out
==939== Profiling application: ./a.out
c[0][0] = 357389824, c[1023][1023] = -714255872==939== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   70.94%  9.1100ms         1  9.1100ms  9.1100ms  9.1100ms  matrixMul(int*, int*, int*, int)
                   15.21%  1.9532ms         1  1.9532ms  1.9532ms  1.9532ms  [CUDA memcpy DtoH]
                   13.85%  1.7782ms         2  889.09us  884.09us  894.08us  [CUDA memcpy HtoD]
      API calls:   94.19%  276.44ms         3  92.146ms  119.31us  276.20ms  cudaMalloc
                    5.11%  15.010ms         3  5.0034ms  1.1188ms  12.758ms  cudaMemcpy
                    0.35%  1.0392ms         1  1.0392ms  1.0392ms  1.0392ms  cuDeviceGetPCIBusId
                    0.23%  678.78us         3  226.26us  219.79us  233.19us  cudaFree
                    0.07%  212.76us       101  2.1060us     310ns  78.656us  cuDeviceGetAttribute
                    0.02%  56.497us         1  56.497us  56.497us  56.497us  cuDeviceGetName
                    0.01%  40.948us         1  40.948us  40.948us  40.948us  cudaLaunchKernel
                    0.00%  2.2660us         3     755ns     377ns  1.5050us  cuDeviceGetCount
                    0.00%  1.2130us         2     606ns     306ns     907ns  cuDeviceGet
                    0.00%     857ns         1     857ns     857ns     857ns  cuDeviceTotalMem
                    0.00%     466ns         1     466ns     466ns     466ns  cuModuleGetLoadingMode
                    0.00%     440ns         1     440ns     440ns     440ns  cuDeviceGetUuid
```



✓ 0s completed at 3:56 PM