GPU 컴퓨팅 Assignment 2

이름 : 이준휘

학번 : 2018202046

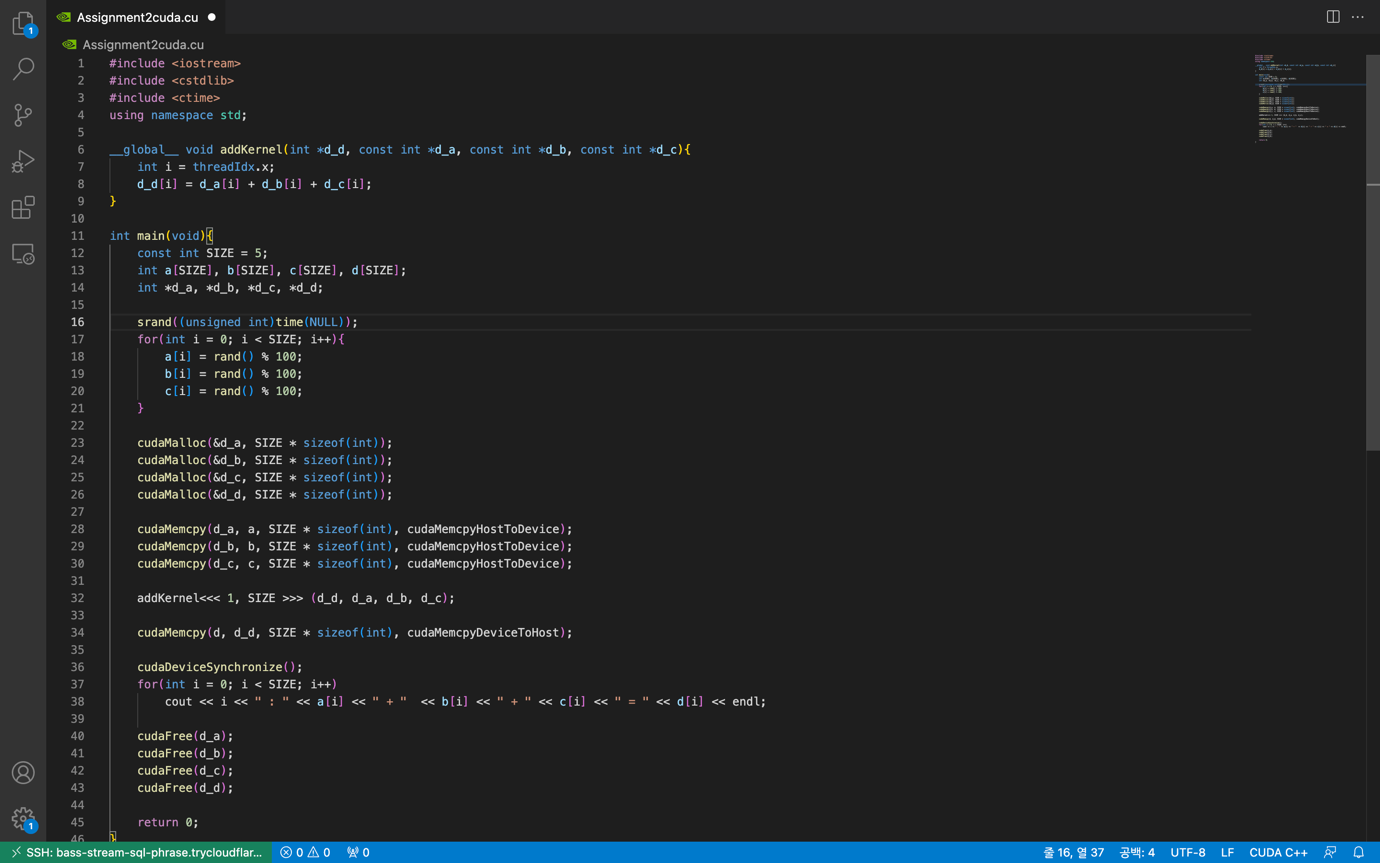
교수 : 공영호 교수님

강의 시간 : 월 수

1. Introduction

해당 과제는 다음 조건에 맞는 코드를 구현한다. Vector의 Size는 5다. Vector 3개를 더하여 이를 d Vector에 저장한다. 이 때 더하는 Vector의 Value는 Random이다.

1. Approach



addKernel() 함수는 \_\_global\_\_ 매개변수를 받아 Host에서 Device에 함수를 수행하도록 명령한다. 해당 함수는 \_\_global\_\_를 사용하기 때문에 return은 void를 사용한다. parameter로 저장할 위치 int \*d\_d와 더할 값 const int \*d\_a, \*d\_b, \*d\_c를 사용한다. Thread의 id값을 threadIdx.x로 받아 이를 i에 저장하고 이를 배열의 인덱스값으로 활용하여 d = a + b + c 연산을 수행한다.

Main 함수는 다음과 같이 진행된다.

Vector의 크기는 5이기 때문에 const int 형태로 SIZE에 5를 할당한다. 또한 해당 값을 바탕으로 int 배열 a, b, c, d를 생성한다. 그 후 Device에서 사용할 pointer를 위한 int \*d\_a, \*d\_b, \*d\_c, \*d\_d를 생성한다.

srand() 함수를 통해 seed값을 현재 시간으로 설정한 후, for문을 통해 a, b, c 배열에 random 값을 할당한다. 할당하는 random value의 크기는 100 미만으로 설정한다.

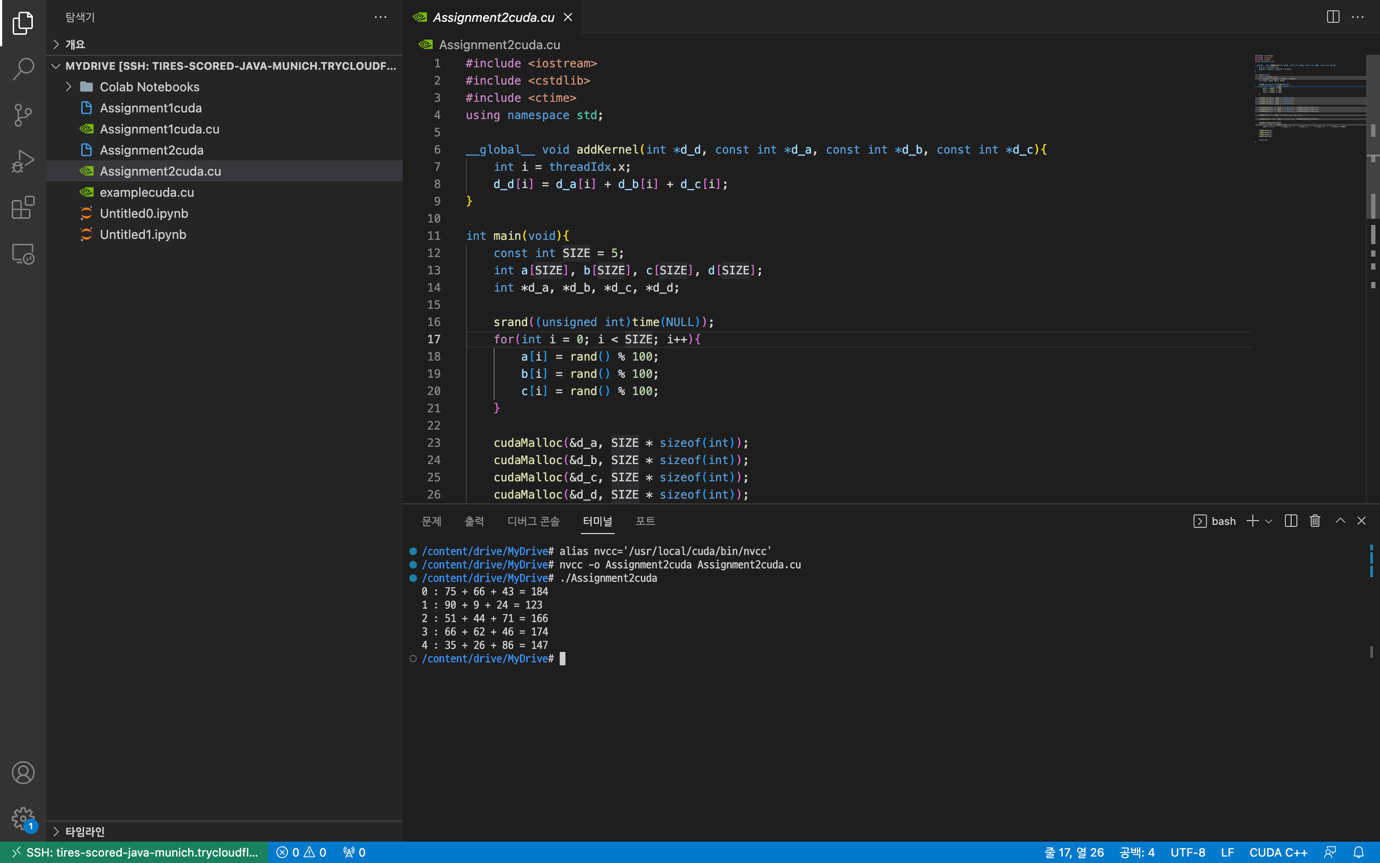
cudaMalloc() 함수에서는 d\_a, d\_b, d\_c, d\_d 포인터에 SIZE(5) \* sizeof(int) 크기의 메모리를 할당한다. 그 후 d\_a, d\_b, d\_c에 a, b, c의 값을 복사하는 cudaMemcpy()를 수행한다. 해당 복사는 Host -> Device임으로 cudaMemcpyHostToDevice 옵션을 추가한다.

addKernel<<< 1, SIZE >>> (d\_d, d\_a, d\_b, d\_c)는 0 ~ SIZE – 1의 ID를 가진 Thread에서 addKernel 함수를 수행한다. 그 후 결과로 나온 d\_d의 값을 d로 옮겨주기 위한 cudaMemcpy() 함수를 수행하며, 이 때는 Device -> Host 임으로 cudaMemcpyDeviceToHost 옵션을 활용한다.

CPU와 GPU의 동기화를 맞추기 위한 cudaDeviceSynchronize() 함수를 수행하며 이후 for문을 통해 연산의 결과를 확인하기 위한 for문으로 결과를 검증한다.

결과를 검증한 후 GPU에서 동적 메모리 할당을 해제하기 위한 cudaFree()함수를 수행하며 이후 프로그램을 종료한다.

1. Result



첫 번째 화면은 Colab을 SSH로 연결하여 해당 프로그램을 컴파일, 수행한 모습이다. 위와 같이 정상적으로 컴파일이 되며, 수행 결과 또한 각 행 별로 연산이 정상적으로 수행된 모습을 확인할 수 있다. 이를 통해 해당 프로그램이 정상적으로 구현되었다는 것을 확인하였다.

1. Consideration

해당 과제를 통해 이전에 의미를 모르고 사용하였던 \_\_global\_\_ 매개변수가 어떤 의미를 지니고 있는지 알게 되었으며 void만 반환해야 한다는 것을 알게 되었다. 또한 thread의 id를 index로 활용하기 위해 threadIdx.x를 사용할 수 있다는 것을 알 수 있었다. 또한 Kernel을 불러올 때 <<<>>>에서 두 번째 인자가 사용할 thread의 개수를 의미한다는 것을 알 수 있었다.

1. Reference

강의자료만을 참고