GPU 컴퓨팅 Assignment 3

이름 : 이준휘

학번 : 2018202046

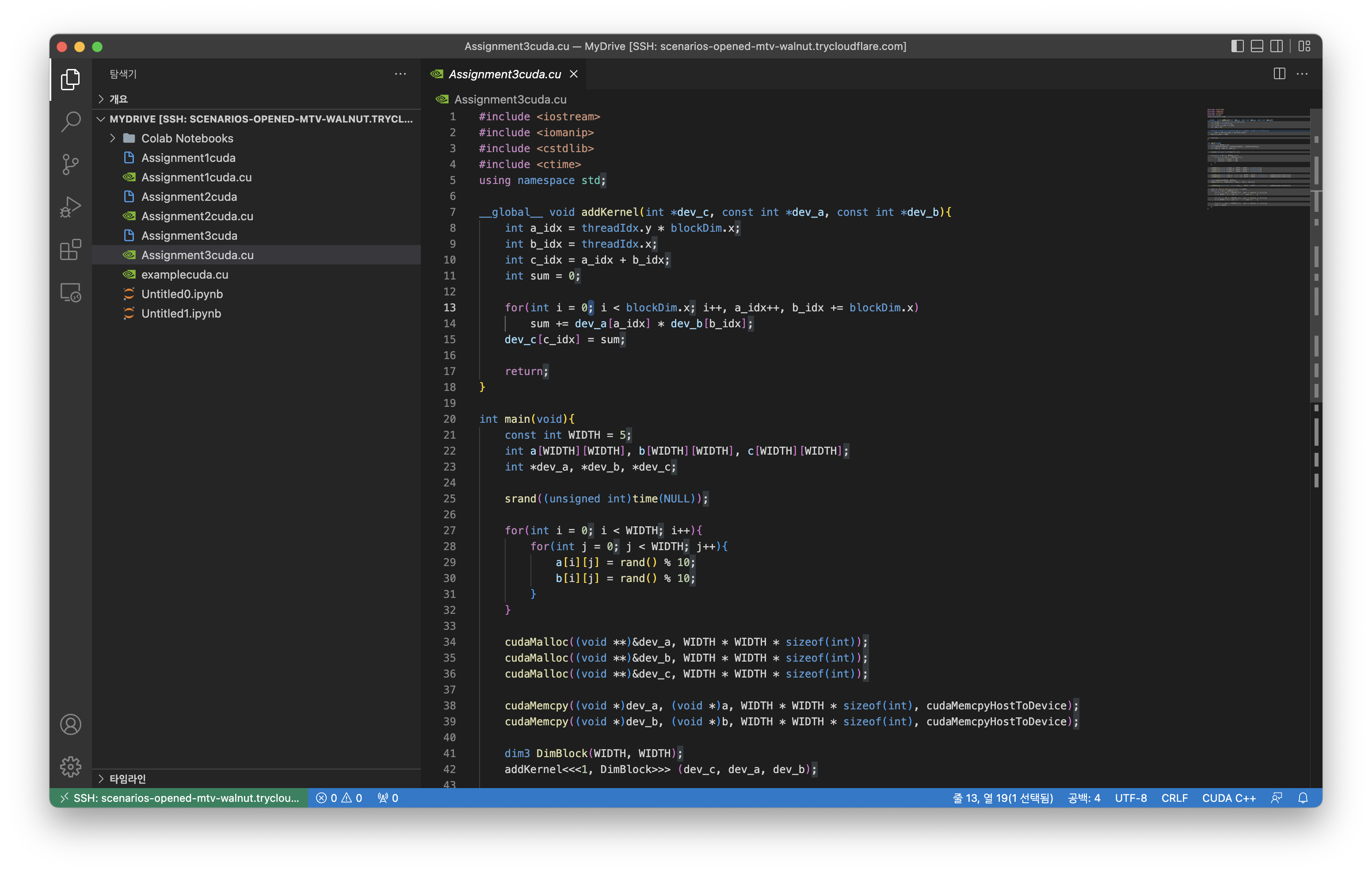
교수 : 공영호 교수님

강의 시간 : 월 수

1. Introduction

해당 과제는 다음 조건에 맞는 코드를 구현한다. 행렬 A와 B간의 행렬 곱 연산을 수행하여 나온 결과를 C에 저장하는 코드를 작성한다. 이 때 행렬의 사이즈는 변수화 하여 유지 보수가 쉽도록 한다. 또한 dim3을 사용하여 2D thread를 사용하도록 한다.

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 \*dev\_c와 연산할 값 const int \*dev\_a, \*dev\_b를 사용한다.

해당 함수에서는 3가지 idx를 사용한다. dev\_a의 idx 값은 행렬 곱 연산에서 특정 행에서 다음 열로 한 칸씩 이동함으로 기준점을 threadIdx.y \* blockDim.x로 잡는다. 이 때 blockDim.x는 행열의 너비로 볼 수 있다. 다음으로 dev\_b의 idx값은 특정 열을 기준으로 행 단위로 움직이기 때문에 threadIdx.x로 설정한다. 마지막으로 더할 곳 c의 idx는 특정 행과 열임으로 a\_idx와 b\_idx를 더함으로써 구할 수 있다.

이후 for문에서는 blockDim.x(WIDTH)만큼 반복하는 코드다. 이 때 a\_idx는 반복 시마다 증가, b\_idx는 blockDim.x만큼 증가시키며 연산을 수행한다. sum에 dev\_a의 idx위치의 값과 dev\_b의 idx위치의 값을 곱한 값을 더한다. 모든 for문의 연산 후에는 해당 값을 dev\_c의 idx위치의 값에 저장한다.

Main 함수는 다음과 같이 진행된다.

행렬의 크기는 const int 형태로 WIDTH에 5를 할당한다. 또한 해당 값을 바탕으로 int 행렬 a, b, c를 생성한다. 그 후 Device에서 사용할 pointer를 위한 int \*dev\_a, \*dev\_b, \*dev\_c를 생성한다.

srand() 함수를 통해 seed값을 현재 시간으로 설정한 후, for문을 통해 a, b 행렬에 random 값을 할당한다. 할당하는 random value의 크기는 10 미만으로 설정한다.

cudaMalloc() 함수에서는 dev\_a, dev\_b, dev\_c 포인터에 WIDTH \* WIDTH \* sizeof(int) 크기의 메모리를 할당한다. 그 후 dev\_a, dev\_b 에 a, b의 값을 복사하는 cudaMemcpy()를 수행한다. 해당 복사는 Host -> Device임으로 cudaMemcpyHostToDevice 옵션을 추가한다.

Device는 WIDH \* WIDTH 개수의 thread를 사용할 예정임으로 dim3 변수 DimBlock 변수의 값을 (WIDTH, WIDTH)로 설정한다. 그 후 addKernel<<< 1, DimBlock >>> (dev\_c, dev\_a, dev\_b)는 2D WIDTH \* WIDTH 크기의 ID를 가진 Thread에서 addKernel 함수를 수행한다. 그 후 결과로 나온 dev\_c의 값을 c로 옮겨주기 위한 cudaMemcpy() 함수를 수행하며, 이 때는 Device -> Host 임으로 cudaMemcpyDeviceToHost 옵션을 활용한다.

텍스트, 스크린샷, 모니터, 검은색이(가) 표시된 사진

자동 생성된 설명

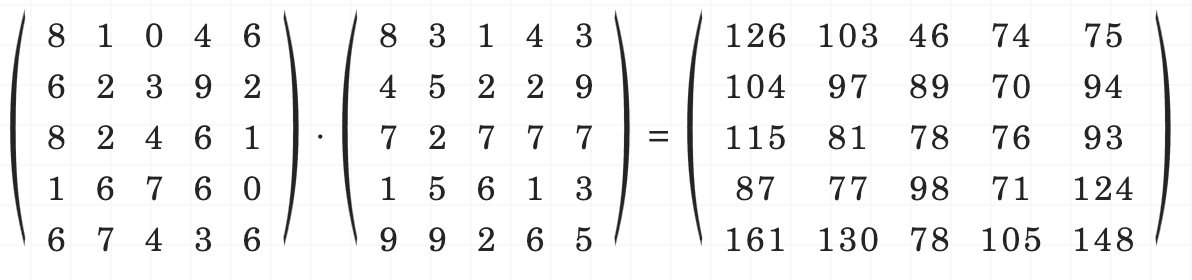
결과를 출력한 후 GPU에서 동적 메모리 할당을 해제하기 위한 cudaFree()함수를 수행하며 이후 프로그램을 종료한다.

1. Result

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

해당 화면은 Colab을 SSH로 연결하여 해당 프로그램을 컴파일, 수행한 모습이다. 위와 같이 정상적으로 컴파일이 되며, 결과가 출력된 것을 확인할 수 있다. 해당 행렬 연산이 정상적으로 수행되었는지 확인하기 위해 해당 값을 행렬 계산기에 넣어서 확인해보았다.



위의 결과와 출력된 결과가 같기 때문에 해당 과제가 정상적으로 동작함을 알 수 있다.

1. Consideration

해당 과제를 통해 2차원 행렬을 1차원 행렬로 변환하여 사용하는 방법을 익힐 수 있었다. 또한 기존에 BlockDim을 숫자로 사용한 것에서 발전하여 dim3라는 변수를 사용하여 thread를 행렬 형태로 사용할 수 있다는 것을 알 수 있었다. 그리고 이러한 thread를 threadIdx의 인자 x, y, z와 blockDim의 인자를 통해 id를 알 수 있음을 공부하였다. 행렬의 연산이 결과와 같이 빠른 시간 내에 수행 할 수 있다는 사실을 통해 이러한 GPU를 사용하였을 때 유용한 연산 형태를 가늠할 수 있었다.

1. Reference

강의자료만을 참고