**Contribution: Yitong Li mainly finish the exercise 1 and Guoqing Liang mainly finish the exercise 2.**

**Code Link：**[CHU-2002/DD2360HT23 (github.com)](https://github.com/CHU-2002/DD2360HT23)

**Exercise 1:**

**1.**

Upload my code: lab3\_ex1\_template.cu to google cloud.

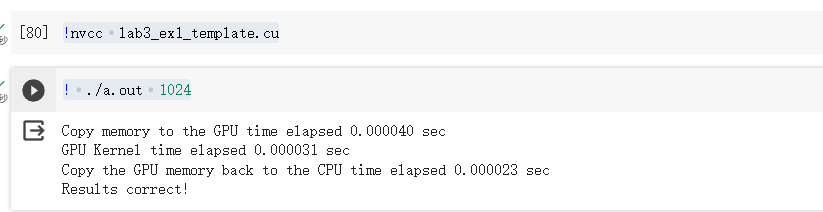
Use the following order to compile:

!nvcc lab3\_ex1\_template.cu

Use the following order to run:

! ./a.out 1024

The format is ./a.out datalength



**2.**

(1). N times in1[i] + in2[i]

(2). Read the in1[i] and in2[i], 2N

**3.**

(1). 1024 threads and (256 + 1024 -1)/256 = 4 blocks

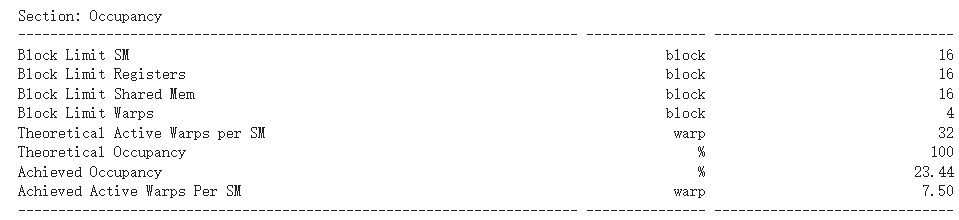
TPB = 256, so the number of block is 4. For each add operation, use 1 thread to calculate.

(2). Follow the order:

!/usr/local/cuda-11/bin/nv-nsight-cu-cli /content/drive/MyDrive/Assignment2/a.out 1024

Get the results.

According to the screenshot:



The Achieved Occupancy is 23.44%

**4.**

(1). It still work just as the following screenshot:

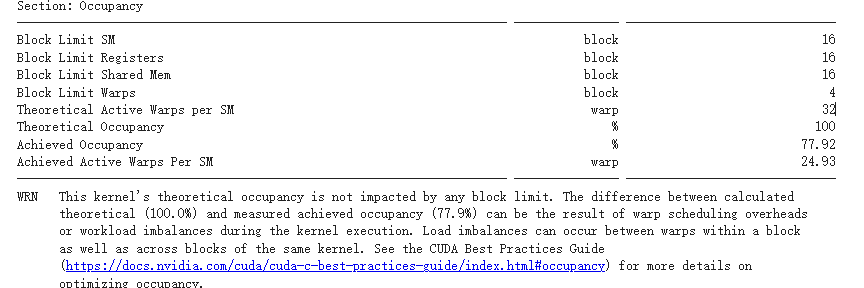


(2).same as the 3.(1)

131070 threads

(256 + 131070 -1 )/256 = 512 blocks

(3).as the screenshot:



The Achieved Occupancy is 77.92%

5.

Test 8 different length in total. The results are following:

|  |  |  |  |
| --- | --- | --- | --- |
| inputlength | data copy from host to device | the CUDA kernel | data copy from device to host |
| 1024 | 0.000233 | 0.000035 | 0.000024 |
| 8192 | 0.000218 | 0.000039 | 0.00008 |
| 65536 | 0.000508 | 0.000038 | 0.000444 |
| 131070 | 0.000838 | 0.000082 | 0.000855 |
| 200500 | 0.001025 | 0.000043 | 0.001344 |
| 1000588 | 0.003731 | 0.000125 | 0.005476 |
| 5008462 | 0.017011 | 0.00049 | 0.025677 |
| 10005765 | 0.035892 | 0.000933 | 0.051195 |

Use it, I drawn the stacked bar chart:

**Exercise 2：**

**1.**

machine learning, graphic processing, data science.

**2.**

Addition: numARows \* numBColumns \* (numAColumns - 1)

Multiplication: numARows \* numBColumns \* numAColumns

**3.**

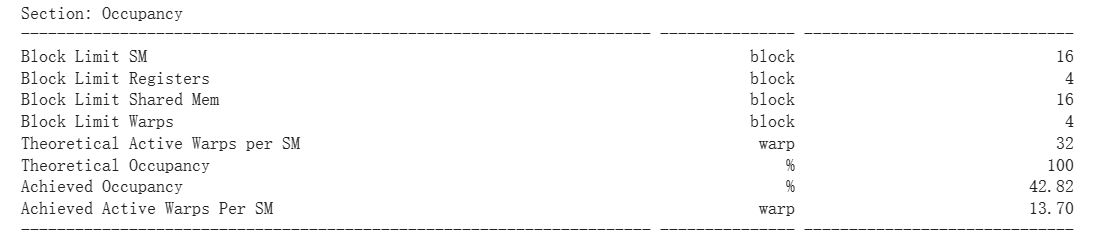
numARows \*numAColumns\*numBRows\*numBColumns

**4.**

(1).Blocks = ((256 + 16 - 1)/ 16) \* ((256 + 16 - 1)/ 16) = 256

Threads = Blocks \* Grids = 128 \* 128 = 16384

**(2).P**



Achieved Occupancy = 42.82%

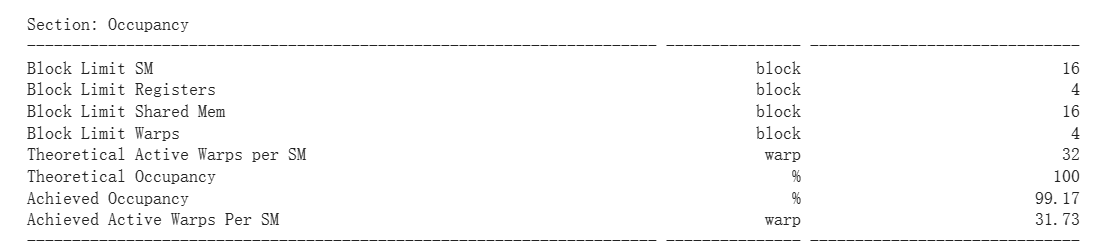
**5.**

(1) Yes

(2).Blocks = ((511 + 16 - 1)/ 16) \* ((4094 + 16 - 1)/ 16) = 8192

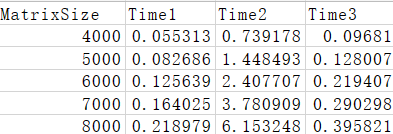
Threads = Blocks \* Grids = 8192 \* 256 = 2097152

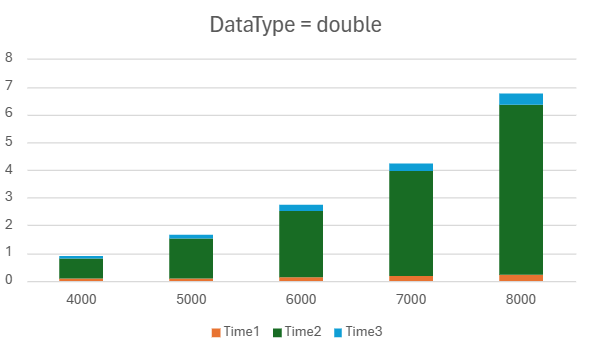
(3).



Achieved Occupancy = 99.17%

**6.**





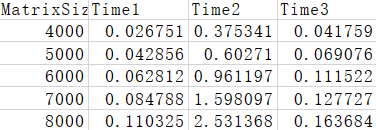
MatrixSize: Rows and Cols for MatrixA and MatrixB

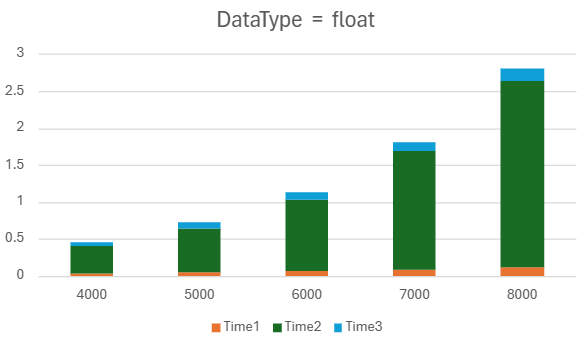
Time1: data copy from host to device

Time2: the CUDA kernel

Time3: data copy from device to host

**7.**





Time consumption for all three steps are much shorter after changing datatype from ‘double’ to ‘float’.