



**Exercise Manual
for
CE/CZ3001
Advanced Computer Architecture**

**Practical Exercises
on
CUDA Programming
for GPU**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY**

CUDA Programming

Learning Objectives

Student will practice programming of GPU using CUDA C in order to appreciate the basic architecture of GPU.

Intended Learning Outcomes

At the end of this exercise, you should be able to

- write basic CUDA programs.
- make use of GPU to solve problem based on parallelism using multi-threading.

Equipment and accessories required

- i) NVIDIA Jetson TX2 embedded board
- ii) HDMI monitor, USB keyboard and USB mouse.

1. Introduction

NVIDIA Jetson TX2 is an embedded GPU-enabled platform designed for AI edge computing. The GPU is based on NVIDIA Pascal architecture with 2 SMs and 256 CUDA cores, while the host is an ARM based multiprocessor (which consists of 2 Denver 64-bit CPUs + Quad-Core A57 Complex).



Fig.1 – Jetson TX2

It runs the Linux operating system, and can be set up as a standalone GPU enabled computing system by connecting a monitor, keyboard and mouse through its various interfaces.

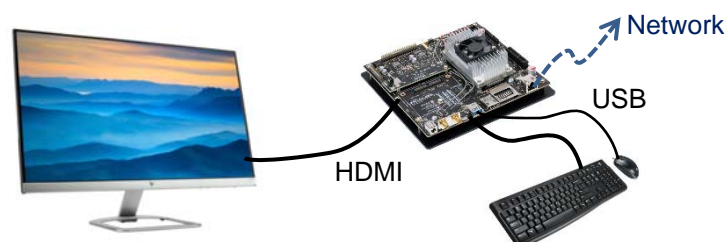


Fig. 2 - Standalone setup of Jetson TX2

CUDA Programming

As such, you should be familiar with some of the basic Linux commands and applications that are commonly used for developing programs in Linux environment. Some examples are as follows:

- know how to open a terminal to enter commonly used commands.
- **ls** command - for listing the (current) directory's contents.
- **cd** command - for changing to another directory.
- using **gedit** application to edit your source code.
- using the **Arrow** keys to repeat previously used commands.
- using **Tab** key to complete a command.
- know how to **compile C program** using gcc – but in this exercise, you will use the NVIDIA **nvcc** compiler instead.

Note: As this is a year 3 course, this exercise 'manual' is purposely written with brevity, such that students will have to learn how to look for relevant information, and more importantly learn how to approach a problem without step-by-step guiding instructions (as commonly found in lower year courses). Nevertheless, most of the exercises had been covered in the lecture (on CUDA programming), and students can refer to the lecture notes (or google) for the detail.

2 Basic CUDA programming

2.1 Hello world!

The following is the customary program when we first explore programming (in C).

```
int main(void){  
    printf("Hello, World - from CPU!")  
    return 0;  
}
```

Modify the code to include two CUDA kernels to produce the output similar to the following when the program is executed.

```
Hello from CPU!  
Hello from GPU1[0]!  
Hello from GPU1[1]!  
Hello from GPU1[2]!  
Hello from GPU1[3]!  
Hello from GPU2[0]!  
Hello from GPU2[1]!  
Hello from GPU2[2]!  
Hello from GPU2[3]!  
Hello from GPU2[4]!  
Hello from GPU2[5]!
```

Discussion:

Observe whether your program produces the output exactly as shown above. What could be the reasons if otherwise?

CUDA Programming

2.2 Vector Addition using parallel threads

$$A = \begin{bmatrix} 22 \\ 13 \\ 16 \\ 5 \end{bmatrix} \quad B = \begin{bmatrix} 5 \\ 22 \\ 17 \\ 37 \end{bmatrix}$$

Given two vectors **A** and **B** shown above, **code a CUDA program that adds the two vectors to produce the result and stored as vector C**. The output of the program should be as follows:

```
A    22 13 16  5
B     5 22 17 37
C    27 35 33 42
```

2.3 Dot Product

For the same vectors **A** and **B** in 2.2 above, **code a CUDA program to compute its dot product**. The output of the program should be as follows:

```
A    22 13 16  5
B     5 22 17 37
Answer = 853
```

3 An application

You are running a webstore that lists the following online items at the respective prices shown.

- (a) 2T harddisk (HD) - \$29.99
- (b) BT earpiece (EP) - \$14.99
- (c) Iphone screen protector (SP) - \$9.99
- (d) 10G USB C thumbdrive (TD) - \$24.99

For one particular week in a particular month, the sale figures for these items are as follows:

Item	Mon	Tue	Wed	Thu	Fri	Sat	Sun
HD	3	2	0	3	4	10	8
BT	5	4	3	5	5	13	11
SP	2	5	3	4	5	21	15
TD	0	1	1	4	3	16	8

Code a CUDA based program that computes the sale amount received for each of the day.

Things to consider: How should you represent the data? How can the program be designed in such a way that it is easily scalable to include many more items as your business expanded?

Hint: The following shows an example of how a 2D array can be defined in C.

```
int 2Darray[2][4]= {
    { 5,  7,  2, 20},
    {15, 11, 22, 18}
};
```