CUDA Programming Task 3

Tazmeen Afroz
Roll No: 22P-9252
Section: BAI-6A

1 Output of all

```
(base) tazmeen@afroz:~/cuda tasks$ chmod +x compile_all.sh
(base) tazmeen@afroz:~/cuda tasks$ ./compile_all.sh
Compiling and running q1.c...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Compiling and running q2.c...
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Compiling and running q3.c...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Compiling and running q4.c...
0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0

Compiling and running q5.c...
0 0 0 0 0 0 0 0 1 2 3 4 5 6 7

Compiling and running q6.c...
0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888

Compiling and running q7.c...
111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0

Compiling and running q8.c...
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Compiling and running q9.c...
111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0

Compiling and running q10.c...
111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444

Compiling and running q11.c...
3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0
```

Figure 1: Output of all questions

Initial run with:

- Position 1: `array[blockIdx.x] = blockIdx.x`
- Position 2: `myHelloOnGPU<<<N, 1>>>(gpuArray);`

Output: `0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15`

2 Question 1

- Position 1: `array[blockIdx.x] = threadIdx.x`
- Position 2: `myHelloOnGPU<<<N, 1>>>(gpuArray);`

Output: `0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0` **Explanation:** Since we're using 1 thread per block, `threadIdx.x` is always 0 for all blocks.

3 Question 2

- Position 1: `array[threadIdx.x] = threadIdx.x`
- Position 2: `myHelloOnGPU<<<1, N>>>(gpuArray);`

Output: `0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15` **Explanation:** We have 1 block with N threads. Each thread writes its ID to its corresponding position.

4 Question 3

- Position 1: `array[threadIdx.x] = blockIdx.x`
- Position 2: `myHelloOnGPU<<<1, N>>>(gpuArray);`

Output: `0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0` **Explanation:** With only 1 block, `blockIdx.x` is always 0 for all threads.

5 Question 4

- Position 1: `array[threadIdx.x] = threadIdx.x`
- Position 2: `myHelloOnGPU<<<1, N/2>>>(gpuArray);`

Output: `0 1 2 3 4 5 6 7 0 0 0 0 0 0 0 0` **Explanation:** We have 1 block with N/2 threads. Only the first N/2 positions are populated.

6 Question 5

- Position 1: `array[threadIdx.x + blockDim.x] = threadIdx.x`
- Position 2: `myHelloOnGPU<<<1, N/2>>>(gpuArray);`

Output: `0 0 0 0 0 0 0 0 1 2 3 4 5 6 7` **Explanation:** We have 1 block with N/2 threads. Each thread writes to positions offset by `blockDim.x` (which is 8).

7 Question 6

- Position 1: `array[blockIdx.x+gridDim.x] = 111*(blockIdx.x+1);`
- Position 2: `myHelloOnGPU<<<N/2, 1>>>(gpuArray);`

Output: `0 0 0 0 0 0 0 0 111 222 333 444 555 666 777 888` **Explanation:** We have N/2 blocks with 1 thread each. Each block writes to positions offset by `gridDim.x` (which is 8).

8 Question 7

- Position 1: `array[blockIdx.x2] = 111(blockIdx.x+1);`
- Position 2: `myHelloOnGPU<<<N/2, 1>>>(gpuArray);`

Output: `111 0 222 0 333 0 444 0 555 0 666 0 777 0 888 0` **Explanation:** We have N/2 blocks with 1 thread each. Each block writes to 2 consecutive positions:

- Even positions get value 111 * (blockIdx.x + 1)
- Odd positions get value 0

9 Question 8

- Position 1: `array[blockIdx.x] = gridDim.x - blockIdx.x - 1;`
- Position 2: `myHelloOnGPU<<<N, 1>>>(gpuArray);`

Output: `15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0` **Explanation:** We have N blocks with 1 thread each. Each block writes a value that's the reverse of its position.

10 Question 9

- Position 1: `array[blockIdx.xblockDim.x]=111(blockIdx.x + 1);`
- Position 2: `myHelloOnGPU<<<N/4, N/4>>>(gpuArray);`

Output: `111 0 0 0 222 0 0 0 333 0 0 0 444 0 0 0` **Explanation:** We have N/4 blocks with N/4 threads each. Each block writes only to the first position of its assigned segment.

11 Question 10

- Position 1: `array[blockIdx.x * blockDim.x + threadIdx.x] = 111*(blockIdx.x + 1);`
- Position 2: `myHelloOnGPU<<<N/4, N/4>>>(gpuArray);`

Output: `111 111 111 111 222 222 222 222 333 333 333 333 444 444 444 444` **Explanation:** We have N/4 blocks with N/4 threads each. All threads in a block write the same value to their respective positions.

12 Question 11

- Position 1: `array[blockIdx.x*blockDim.x + threadIdx.x] = blockDim.x - threadIdx.x - 1;`
- Position 2: `myHelloOnGPU<<<N/4, 4>>>(gpuArray);`

Output: `3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0` **Explanation:** We have N/4 blocks with 4 threads each. Each thread writes a value calculated as:

- `blockDim.x - threadIdx.x - 1` = `4 - threadIdx.x - 1` = `3 - threadIdx.x`
- For threadIdx.x = 0, 1, 2, 3, this gives values 3, 2, 1, 0
- This pattern repeats for each block