RHODES UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

EXAMINATION: JUNE 2017

COMPUTER SCIENCE HONOURS
PAPER 3 – GPU PROGRAMMING

Internal Examiner: Prof. K. Bradshaw

MARKS: 120
DURATION: 2 hours

External Examiners: Prof. M. Kuttel

GENERAL INSTRUCTIONS TO CANDIDATES

1. This paper consists of 8 questions and 6 pages. ***Please ensure that you have a complete paper.***
2. Answer all questions and please number your answers clearly.
3. State any assumptions and show all workings. Calculators may be used.
4. Diagrams are encouraged and should be labelled.
5. Provide answers that are concise, legible and clearly numbered.
6. Answer all questions in the answer book provided.
7. You may bring into the examination venue the printed copy of the code handed out to you 24 hours ago with appropriate annotations, provided this has been checked and signed by the internal examiner prior to being seated.
8. The Concise Oxford English Dictionary may be used during this examination.

PLEASE DO NOT TURN OVER THIS PAGE UNTIL TOLD TO DO SO.

Question 1 **(20 marks)**

In the context of CUDA, provide a very brief definition of each of the following terms:

a) Branch divergence
b) Data parallelism
c) Kernel
d) Dynamic parallelism
e) Instruction latency
f) Memory fence
g) Pinned memory
h) Streaming multiprocessor
i) Loop unrolling
j) L1 cache

Question 2 **(5 + 3 = 8 marks)**

a) Describe the SIMT architecture classification with reference to Flynn's taxonomy.

b) Explain Amdahl's law.

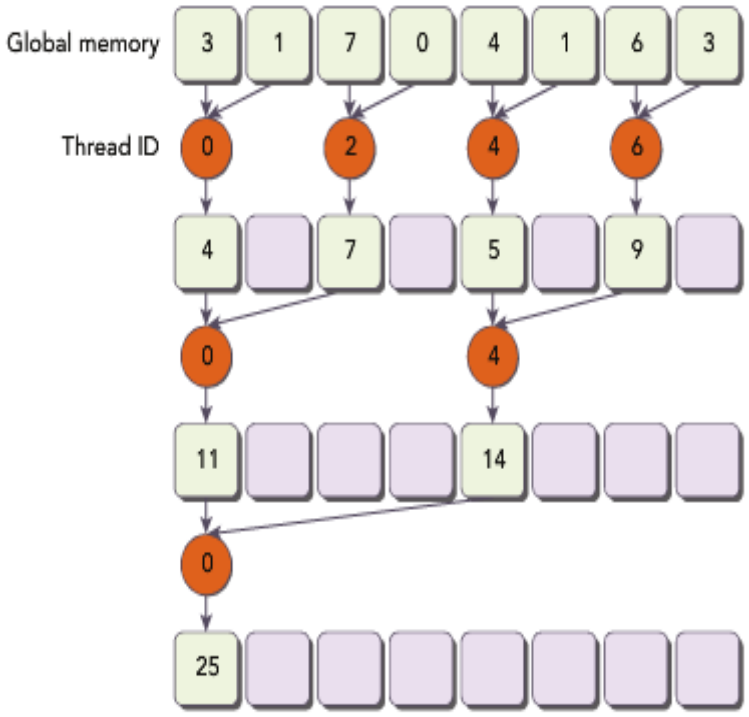
Question 3 **(4 + 8 + 2 = 14 marks)**

GPUs typically provide the greatest speedup benefits when processing very large scale datasets as long as all cores can be used effectively.

a) Define occupancy and state why this metric is important in GPU programming.
b) Explain how achieved occupancy is affected by the grid and block dimensions of a kernel. Hint: it may be worthwhile briefly discussing the warp execution model.
c) What two GPU hardware limitations affect achieving optimal occupancy?

Question 4 **(4 + 8 = 12 marks)**

Consider the illustration below showing a naïve implementation of a parallel reduction using the sum operator.

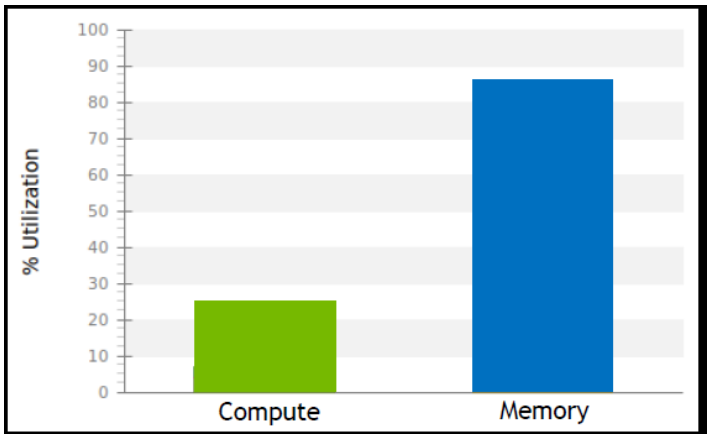


a) Focussing on branch divergence, explain why this implementation is considered inefficient.

b) Describe two ways of improving this implementation. Be sure to mention what aspect of performance is being improved. You may use diagrams to illustrate the changes.

Question 5 **(2 + (4 x 3) = 14 marks)**

While profiling a kernel, NVIDIA's visual profiler, *nvpp*, returns the following graph:



a) Is the kernel compute- or memory-bound? What does this mean to overall kernel performance?

b) Without any knowledge of what the kernel was programmed to do, mention four optimizations you might consider to remedy this situation and potentially improve the overall execution of the kernel.

Question 6 **(10 marks)**

Consider the data definitions and kernel code given below for manipulating points on a 2-dimensional map, represented by x- and y-coordinates.

```
#define LEN 1<<20
struct point { float x; float y; };
point mymap[LEN];

__global__ void testmap (point *data, point *result, const int n)
{ unsigned int i = blockIdx.x * blockDim.x + threadIdx.x;

  if (i < n)
     { innerStruct tmp = data[i];
       tmp.x += 10.f;
       tmp.y += 20.f;
       result[i] = tmp;
      }
}
```

Explain what optimizations you would make to the above code to improve performance. Be sure to give reasons for your choices.

Question 7 **((3 x 4) + 4 = 16 marks)**

To achieve high memory bandwidth, shared memory is divided into 32 equally-sized memory modules, called banks. However, due to bank conflicts, poor effective bandwidth is sometimes achieved.

a) Describe three typical scenarios that can occur when a request to shared memory is issued by a warp. For each of these scenarios, explain under what conditions this situation might occur, and what the penalty to bandwidth would be. You may use examples to illustrate your answers.

b) Describe one way to avoid bank conflicts.

Question 8 **(4 + 8 + 8 + 6 = 26 marks)**

Consider the two CUDA kernels for copying and transposing matrices, the code for which was handed to you 24 hours prior to this exam with the suggestion that you profile it (with L1 cache enabled) to understand the execution efficiencies. You should have pages 5 and 6 of this printout in your possession.

a) Define theoretical peak bandwidth and effective bandwidth.

b) The *copyRow* kernel runs significantly faster than the *copyCol* kernel. Referring to the effective bandwidth values (as well as other metrics you consider relevant) and memory access patterns discussed in class explain why this is so.

c) Explain why there are differences between the values for global load/store throughput obtained for the two transpose kernels (i.e. *transposeNaiveCol* and *transposeNaiveRow*). Hint: your answer to (b) should be relevant to this question as well.

d) How would you optimise (at least two changes need to be considered) one of the transposition kernels to achieve greater effective bandwidth and faster execution? Give reasons for your suggestions.

END OF EXAMINATION