BCIT® BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY

COMP 8551
Fall 2018 – Instructor: Borna Noureddin
Assignment #3

All work should be done in pairs.

Total marks: 100

CHOOSE ONE OF THE FOLLOWING SETS OF QUESTIONS TO ANSWER

The code must be written in Objective-C, C, C++ or ARM/NEON assembly instructions for SET 1, and C++ or Intel MMX/SEE assembly instructions for questions SET 2. All files required to build and deploy the app/executable must be provided. Submit all your project files and any documentation to the D2L dropbox folder as a single ZIP file with the filename A00ABC_A00XYZ_Asst3.zip, where ABC and XYZ are your A00 numbers. All required documentation (README file, code comments, etc.) must be included.

SET 1 (ARM/NEON)

a) [35 marks] Start with the SomeNEONTests.zip XCode project. The app lets the user select a photo from their photo library and converts it to a grayscale image. The convertToGrayscale() function performs the grayscale transformation in C++. Implement the convertToGrayscaleNEON_intrinsics() function to perform the same function using NEON intrinsics. Note that you must run this code on an ARM-processor-equipped device, such as an iPod touch, iPhone or iPad. The simulators use an x86 instruction set, and the code will not compile if targeted for the simulator. How does the performance of the NEON code compare with the C++ code?

b) [25 marks] Now implement the convertToGrayscaleNEON_asm() function to perform the same function using NEON assembly code instructions. How does the performance compare with the C++ code and the NEON intrinsics code? Which is the most efficient?

c) [40 marks] Create an OpenGL ES app that performs a simple transformation on a cube or other simple object (you can use one of your previous assignments). Implement the matrix multiplication code in ARM NEON code (you can use intrinsics or assembly code). Compare the performance of the NEON code with the GLKMatrix4Multiply() code. Can you achieve better performance? If so, by how much, and if not, why not?

SET 2 (x86/x64)

a) [50 marks] Start with the sample code given in class for x86/x64 assembly language. In one of them, the windows application reads in a colour image and a “kernel” image. The kernel image is “blended” with the original image according to the following:

```
pBlendImg[i][j] = pOrigImg[i][j] * blendFac + pKernelImg[i][j] * (1 -
blendFac);
```

where **pOrigImg** is the original image, **pKernelImg** is the kernel image, **pBlendImg** is the resulting image, and **blendFac** is the blending factor. Describe how the blitBlend function works for each of the three simMode’s. Include descriptions of how each of lines of code with the intrinsics and asm instrunctions works. Which mode runs the fastest and second fastest, and why?

b) [30 marks] The code uses the alpha channel of the kernel image as the value for blendFac. Modify the code in both cases to use the alpha channel of the destination image for the blending factor instead.

c) [10 marks] Write a simple function that takes an object as an argument by value. Write code to call the function and show the disassembly code associated with the function call. Now write the same function, but have it take the argument by reference, pass the object by reference and show the new disassembly code. What is the difference?

d) [10 marks] Write a simple function that takes no arguments and performs a simple math function. Show the disassembly code associated with the function. Now make the function inline and show the new disassembly code. What is the difference?