COEDS4 – LAB #7 REPORT

GROUP 18

Khalid Asad 1147299 Vishal Kharker 1140837

asadk@mcmaster.ca; kharkevp@mcmaster.ca

March 28, 2016

Exercise 1:

In order to get a median value from *experiment1* there were very minor changes that needed to be implemented. In order to get the median from both the arrays, we simply used a bubble sort in the find_min_max_0 and find_min_max_1 functions. This helped us get the values from the array, sort them out in an ascending order then we divided the array in half. This marks the median for an array stored at data_array[ARRAY_SIZE/2].

Exercise 2:

We created seven blocks for the memory partition. We used a block size of 32x32 and all seven arrays were declared as int. We initially tried with 4x4 matrices, then 8x8, and 16x16. All of these resulting matrices were confirmed with the online matrix calculator to be correct. However, with 32x32 matrices, we seemed to run out of memory, so the resulting matrix would only have the top left corner with the correctly calculated values. Everything after that had values, they were just not correct when compared with the matrix calculator online. We spent over 10 hours just trying figure this last part of this exercise out, but in the end we could not. Some possible solutions could have been to reduce the amount of blocks and arrays being used to 6 or less. Another solution could have been to scrap the memory partition method in favour of structures.

Exercise 3:

In this exercise we attempted to read images one at a time, run the filter on the image and then output a final image. This part of the take home outputs a final image for image1.BMP and reads image2.BMP but it does not output and further images. The error has to do with Readque and Writeque.

Exercise 4:

In this exercise CPU1 handled the bottom half and CPU2 handled the top half of the calculations for the final matrix. Half of ArrayA and all of ArrayB was passed into CPU2. We used 10 flags to handle the states of each CPU. If we decide to generate the array again and run the calculations, it works over and over again. When we included code for the performance counter, the output would either be nothing or symbols. The performance counter implementation would

have been simple as we have already done it in exercise 2. Amin and another student were having the same problem in the in-lab portion for lab 8. He could not figure out the problem.

NOTE: the submission of this lab was about an hour late because when copying the files for exercise 4, the cpu2.c file got overwritten, and we had to scramble to rewrite that part of the code. Fortunately, cpu1.c and cpu2.c are pretty similar, so it took us only an hour.