Dr. Tolga Soyata. © University at Albany, Dept. of Computer Engineering, tsoyata@albany.edu

ICEN360-GPU GPU Architecture and Programming

Project 4 (25pts) Due: Nov 15, 2016 11:59PM

- **You will be writing this project on our GPU cluster. Your code is expected to work in the cluster.**
- **If you do not allocate/de-allocate memory properly, points will be taken off your program. So, your malloc() and cudaMalloc()'s must be used properly in your code.**
- **Each late day will subtract 2 points from your assignment points**

- **PART A (8 points):** Take the CPU image rotation program and write a GPU version of it.
- The program will rotate a given image file, as specified in the command line.
- The program will create **N** images, each of which is a 360/N degree rotation of the original image.
- Use only global memory in this PART A.
- Submit your code as **imrotateG.cu** through Blackboard, along with everything needed to compile it (ImageStuff etc.). Call it **Proj4PartA.zip**.
- Time your code for the execution of all **N** rotations, start to finish. Do not include the I/O time or the image transfer time from CPU to GPU in your timing.
- The program is launched with the following command line parameters:
  - **./imrotateG Infile Outfile N**
  - For example: **./imrotateG Astronaut.bmp AROT.png 9** will read the Astranaut.bmp file, and write 9 different output files named AROT1.png, AROT2.png, ... AROT9.png.
  - AROT1.png is the original file (0° degree rotation), AROT2.png is the 40° rotated version of the image, AROT3.png is the 80° rotated version, ..., and AROT9.png is the 320° rotated version.
  - Check to make sure that the user did not specify **N** to be higher than 30.
  - After this is done, you can go to the command line and use ImageMagick to create an **animated GIF** out of all these 9 files with the file name **AROT.gif**.
  - If you would like to turn the results into an animated gif, use this command:
  - **convert AROT*.png AROT.gif**
  - Note: this could result in a huge, slow file depending on your input. You may want to use a flag like **-resize** to scale it down.

- **PART B (8 points):**
  - Now, rewrite the same program to use Shared Memory.
  - Submit your code as **imrotateGS.cu** through Blackboard, along with everything needed to compile it (ImageStuff etc.). Call it **Proj4PartB.zip**.
  - Try to take advantage of the fact that, you are computing almost the same things **N** times and shared memory could substantially help you in this case.
  - The performance of your code will not matter in PART B. Use the very first implementation that comes to your mind.
  - What matters in the PART B is the fact that, you re-engineered PART A to substantially take advantage of shared memory

UNIVERSITY AT ALBANY
State University of New York

- **PART C (9 points):**
  - Take your code **imrotateGS.cu** and analyze its occupancy using the CUDA occupancy calculator. Choose different block sizes, shared memory sizes and play with any parameter you think is important. Rerun the code with multiple different configurations and report your results. You are trying to improve upon your very initial implementation by using the CUDA Occupancy Calculator.
  - Use the Profiler to detect potential bank conflicts or thread divergence and re-engineer your program to prevent those.
  - Submit 4 of the best different kernels/configurations you tried as **imrotateGS2.cu**, in addition to the very first inefficient one**.** Allow running these 5 total configurations from the command line as follows:
  - **./imrotateGS2 Astronaut.bmp AROT.png 9 4**
  - where the newly introduced command line parameter "4" is the new "configuration" option. "1" will be the very first kernel you tried in your PART B with some default block size, etc. 2, 3, 4, 5 are incremental improvements which may launch completely different kernels, the same kernel with different shared memory size, etc.
  - Your program **imrotateGS2.cu** should be reporting the run times of the kernel.

Submit a report called **Proj4Report.pdf** that briefly describes what you tried and the screen shots of the Profiler as well as the CUDA Occupancy calculator for the different options you tried. Also, the runtimes of each idea you tried and tested with the CUDA Occupancy Calculator and Profiler should be reported in a Table.