University of Cape Town, Computer Science

CSC2002S, Computer Architecture Assignment

15 marks

Image processing is the manipulation of images. Raster images comprise of units called pixels. In a 3-channel colour image, each pixel, in this case, comprises of a red (R), green (G) and blue (B) value. Each of the colour values is a byte wide (each pixel requires 3 bytes). The combination of these primary colours is what allows us to display different colour variations, e.g.:

Colour	R	G	B
Red	255	0	0
Green	0	255	0
Blue	0	0	255
Yellow	255	234	0
White	255	255	255
Black	0	0	0

Greyscale images only require one channel per pixel (each pixel requires 1 byte), where 255 indicates white, 0 indicates black and the values between are varying shades of grey.

Images can be stored in many different formats, mostly in binary form. However, for debugging purposes, storing images in ASCII form is useful. Portable Pixel Map (PPM) can store images in ASCII form. Here is an example of a PPM file:

Line	**Text**	Comment
1.	**P3**	Type of PPM file
2.	**# Jet**	File comments
3.	**64 64**	Number of rows and columns in the image
4.	**255**	Maximum value possible
5.	**166**	R value for pixel 1
6.	**186**	G value for pixel 1
7.	**181**	B value for pixel 1
8.	**175**	R value for pixel 2
9.	**187**	G value for pixel 2
10.	**190**	B value for pixel 2
…	…	…
12290.	**150**	R value for pixel 4096
12291.	**175**	G value for pixel 4096
12292.	**165**	B value for pixel 4096

Lines 1 to 4 form the header of the PPM file. They provide information about the image. Line 1 refers to the type of PPM file. In this case, "P3" means that the file is a colour image stored in ASCII form ("P2" would mean an image is greyscale stored in ASCII form).

Comments for the image can be stored using the # symbol. The image above has a single comment in line 2. For simplicity, we will just use a 3-letter word to describe the image as a comment.

The image size is stored in line 3. The first 64 refers to the number of rows in the image. The second 64 refers to the number of columns in this image. Multiplying the two values will provide you with the total number of pixels in the image ($64\ rows * 64\ columns = 4096\ pixels$).

The remainder of the file consists of RGB values for each pixel, in **ASCII form**. Each line consists of 1 value, which will either be an R, G or B value for a pixel.

Here is the PPM of a greyscale file:

Line	**Text**	Comment
1.	**P2**	Type of PPM file
2.	**# Jt2**	File comments
3.	**64 64**	Number of rows and columns in the image
4.	**255**	Maximum value possible
5.	**177**	Value for pixel 1
6.	**184**	Value for pixel 2
7.	**187**	Value for pixel 3
8.	**188**	Value for pixel 4
9.	**188**	Value for pixel 5
10.	**190**	Value for pixel 6
…	…	…
4098.	**159**	Value for pixel 4094
4099.	**139**	Value for pixel 4095
4100.	**163**	Value for pixel 4096

The structure in the greyscale PPM is almost identical to the colour one. The only difference in the header is the type of file has changed in line 1 (P2). Note how each pixel only requires one line, whereas in the colour PPM, each pixel required three lines.

Question 1

Write a MIPS program (**increase_brightness.asm**) that will read in a colour PPM and increase each RGB value by 10. If the addition of 10 increases the value over 255, store the value as 255. This new image should be written to a new file. After execution display the average of all RGB values of the original file, as well the average of all RGB values of the new file, as a double value on the console, e.g.:

```
Average pixel value of the original image:
0.72084067350898684
Average pixel value of new image:
0.76005635978349673
```

Question 2

Write a MIPS program (**greyscale.asm**) that will read in a colour PPM P3 image and convert this to a greyscale PPM P2 image. A greyscale pixel value is calculated by finding the average of its RGB values. Decimals are rounded down to the nearest whole number, e.g.: for a pixel with RGB values of 166, 186 and 181 respectively, the greyscale pixel value will be 177. You will need to change the file type in the header of the new greyscale file to “P2”.

Points to note:

1. You may display PPM files with a image editor such as GIMP. You may view the characters in PPM files using a text editor.
2. You may assume the input image provided will always be 64x64 pixels in size.
3. You may assume the input image header will always take the form of the example provided above.
4. You may hardcode image headers, within your program.
5. You may hardcode file names within your program – just be sure to explain usage in your readme file, see point 9. They should be absolute paths, not relative ones.
6. MIPS services 13 and 14 must be used separately, i.e., open the file with 13 and then read it with 14 or write to it with 15. Always close all files when done with service 16.
7. The RGB values will be in the range [0, 255]. Numbers are not padded, i.e., numbers less than 3 digits long will **not** be padded with zeros to make them 3 digits long.
8. QtSpim uses the [LF] character to mark end of lines (view your data memory after reading files to verify this). Your OS, when writing files, may differ. Linux uses [LF]. Mac uses [CR]. Windows uses both [CR][LF] together.
9. Your code should be **appropriately commented**, and you should **make regular use of git**.
10. You **must provide a readme file** with instructions on how to run your code, i.e., where in your code to specify filenames, etc).
11. Submit your solutions as a compressed archive to the assignment post. This should contain your solution for both questions, a readme file and a git repository.

Marking Guide

Description	Marks
Question 1	
Input file is read correctly	1
ASCII to integer conversion implemented correctly	1
Increased RGB values by 10	1
Integer to string conversion implemented correctly	1
Output file is written correctly; can be opened with an image editor/viewer	2
Correct averages outputted	1
Question 2	
Input file is read correctly	1
ASCII to integer conversion is implemented correctly	1
Correct greyscale calculation	2
Integer to string conversion implemented correctly	1
Output file is written correctly (with the correct header); no extra ASCII values present in the file; file can be opened with an image editor/viewer	3
Total	**15**
Code does not run/execute or produces an exception	-3
No readme file	-1
No git (with at least 5 commits over 2 days)	-1
Late penalty (10%). No late submissions after one day late	-1.5