Computer Architecture Homework #1

Due 2019/10/15 13:00 Tuesday (CEIBA, no late homework is allowed.)

(30%) Part I: Implement a Sort Procedure (P & H, Section 2.12) using QtSPIM.

In class, we introduced MIPS instruction set and showed you an example of sort procedure in instructions. Now we ask you to implement sort procedure using QtSPIM, a MIPS processor simulator which support pseudo-instructions. Repeat the sorting program on page 132 to sort 10 numbers {-1, 3, -5, 7, -9, 2, -4, 6, -8, 10}
You have to modify ***HW1_p1.s*** for your implementation. Snapshot the console window and saved it as ***HW1_p1.jpg***. Both HW1_p1.s and HW1_p1.jpg have to be submitted.

(70%) Part II: Denoising salt-and-pepper noise using the median filter.

In the previous part, you have written a code that can sort a series of numbers. Now it can be applied to an application requiring a median filter.

Salt-and-pepper noise is impulse noise caused by sharp and sudden disturbances in image signals. Since the noise makes pixels either white or black, denoising process can be done with simply a median filter. For a 3-by-3 2D array. Applying the median filter is to sort all 9 elements in the array, and then replace the center element with the median of the sorted numbers while keeping other elements unchanged. The following example shows how the median filter works.

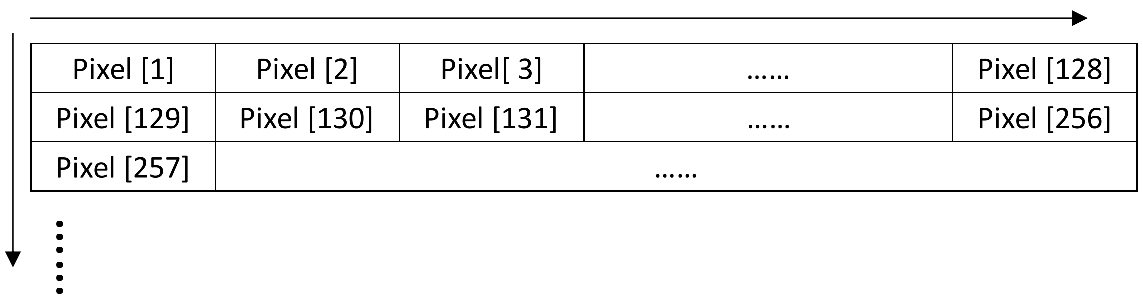
7	5	3
1	0	4
6	2	8

$$\xrightarrow{\text{Sorting : 0, 1, 2, 3, 4, 5, 6, 7, 8}}$$

7	5	3
1	4	4
6	2	8

In this part, you are asked to apply a 3-by-3 median filter on a given 128-by-128 image with salt-and pepper noise. To reduce calculation complexity, for your convenience, the median filter is only applied to the original input array, which means the previously updated pixel values do not affect the latter filtering results. Also, we do not need to deal with boundary conditions. It means that you do not need to calculate the values when the center of the filter is on or out of the boundaries. Therefore, your final result can be represented by a 126-by-126 array.

You have to modify ***HW1_p2.s*** for your denoising implementation. Here we provide you the input data and the output printing format. The input data are the pixels as arranged below.



Pixel [1]	Pixel [2]	Pixel[3]	……	Pixel [128]
Pixel [129]	Pixel [130]	Pixel [131]	……	Pixel [256]
Pixel [257]	……			

You have to store the console to ***ans.txt*** for our convenience to verify your correctness. You may use ans.txt with the help of any programming language (C/C++, python, Matlab etc.) you are familiar with to check the output (126*126) image.

Note:

1. We recommend you directly use your sort function in Part I, but should be careful about the usage of registers.
2. We also provide you two smaller array (7*7 & 15*10) as inputs for you to check your correctness when writing your code.
3. If you have trouble in Part 2, you can write a detailed ***report.pdf*** to get partial credit.

Related Material:
Refer to 20191001_CompArch_SPIM_Simulator.pdf on CEIBA.
A.9 SPIM (P & H, p. A-40).

Submission:

- Upload the results to CEIBA
- Your work should be submitted in a compressed file following the naming convention, CA1081_yourID.zip (for example, CA1081_b06901999.zip). The file should include:
  - HW1_yourID/
    - HW1_p1.s , HW1_p2.s (the SPIM source code)
    - HW1_p1.jpeg
    - ans.txt (for correctness checking of Part II)
    - report.pdf (if you can't finish ans.txt)
    - README.txt (if you think you need it)