Computer Architecture Homework #1

Due 2020/10/27 13:00 Tuesday (CEIBA, no late homework is allowed.)

(40%) Part I: Implement a Sort Procedure using RARS.

In class, we introduced RISC-V instruction. Now we ask you to implement sort procedure using RARS, a RISC-V processor simulator which support pseudo-instructions. Sort 10 numbers {-1, 3, -5, 7, -9, 2, -4, 6, -8, 10} You have to modify ***HW1_1.s*** for your implementation. Snapshot the console window and saved it as ***HW1_1.jpg***.

(60%) Part II: Salt-and-pepper noise denoising with median filter

In the previous part, you have written a code that can sort a series of numbers. Now it can be applied on some applications.

Salt-and-pepper noise is impulse noise caused by sharp and sudden disturbances in the image signal. Since the noise is either white or black pixel, denoising process can be done with simply a median filter. For a 3-by-3 2D array, the median filter sorts all 9 elements in the array, and then replace the center element with the median of the sorted numbers but keep other elements unchanged. The following show an example about how the median filter works.

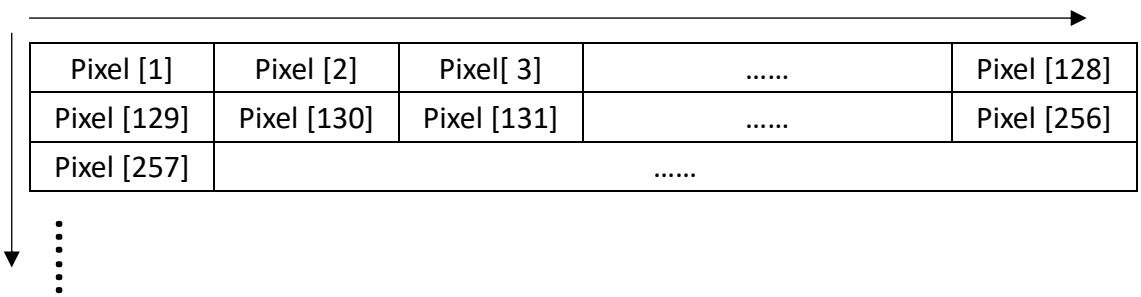
7	5	3
1	0	4
6	2	8

Sorting : 0, 1, 2, 3, 4, 5, 6, 7, 8

7	5	3
1	4	4
6	2	8

In this part, we ask you 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 convenience, the median filter only works on the original input array, which means the previously updated pixel value do not affect the latter filtering result. Also, we do not consider boundary condition. That means you don't need to calculate the value when the center of the filter is on or even out of the boundaries, so our result will be a 126-by-126 array.

You have to modify ***HW1_2.s*** for your denoising implementation. We already provide you the input data and the output printing format. The input data stores pixels from top to down and left to right as showed below.



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

You have to dump the memory to ***memory.txt*** (See note 4.) for our convenience to verify your correctness. You may check your answer with the help of any programming language (C/C++, python, Matlab etc.) you are familiar with.

Note:

1. We recommend you directly use your sort function in the previous part, but should be careful about the use of registers.
2. We also provide you two smaller array (7*7 & 15*10) as input for you to check your correctness easier when writing your code.
3. If you have trouble in Problem 2, you can write a ***report.pdf*** with detail to get partial credit!
4. File → Dump Memory → Memory Segment(.data) → Text/Data Segment → Dump to File… → memory.txt

Related Material:
RARS: https://github.com/TheThirdOne/rars

Submission:

- Upload the results to CEIBA
- Your work should be submitted in a compressed file following the naming convention, HW1_yourID.zip (for example, HW1_b07901999.zip). The file should look like: **(5% penalty for wrong format)**
  - HW1_yourID.zip
    - HW1_yourID/
      - HW1_1.s , HW1_2.s (The assembly code)
      - HW1_1.jpg
      - memory.txt
      - report.pdf (If you can't finish memory.txt)
      - README.txt (If you think you need it)