CS 3340 Computer Architecture – Spring 2019 – Mazidi

Homework 6: MIPS FP Operations

Worth 200 points

Objective: Gain more experience with MIPS programming in the context of a more complex problem.

Big Picture:
- read the input file "input.txt" into a buffer in memory
- extract the string "numbers", convert them to integers and store in an array
- print the integers to console
- sort the integers in place using selection sort
- print the sorted integers to console
- calculate the mean, median, and standard deviation, printing results to the console

Instructions

The following lists and describes the functions you should write and call from the main program.

1. Write a function to read the text from the input file 'input.txt' (on Piazza) and place it in a buffer. A buffer of 80 bytes is more than enough. Before you call your function, set $a0 equal to the address of the filename and $a1 to the address of the buffer where data is stored. The function should return the number of bytes read in $v0. In the main program, print an error message and terminate the program if $v0 <= 0.
2. Write a function to extract integers from the text input buffer and store them in an array of 20 words. Before the function, set $a0=address of the array, $a1=20, $a2 the address where the buffer starts.

   Helpful hints: The input buffer is just a series of ASCII bytes. You will loop through it byte by byte. As you load a byte from the buffer, ignore the byte if it is <48 (ASCII for 0) or >57 (ASCII for 9). If it is within this range, it is a digit. Subtract 48 to convert it from ASCII to int. Multiply the register you are using as an accumulator by 10, then add this new digit. When you load in a byte that is equal to 10, this is newline, so you are done with the integer you were converting; save the integer to the next array element. When you load a byte that is equal to 0 you have reached the end of the data.

3. Write a function to print the array of ints as shown in the sample output below. You will print the array before you call the sort, and after it is sorted, with appropriate text messages.
4. Write a function to sort the array by a selection sort. You can use the algorithm from any of your textbooks or Wikipedia: https://en.wikipedia.org/wiki/Selection_sort

In the remaining functions, before calling them, set $a0 to be the start of the array and $a1 to be the length of the array. Return integer values in $v0 and float values in one of the $f registers.

5. Write a function to calculate the mean. Use single precision for the mean. Store the mean in memory as a float.
6. Write a function to calculate the median. Write the function so that if the length is odd, it returns the middle value as an integer. If the length is even, average the two middle values and return the median as a float. Set $v1 to be a flag to indicate whether the result was int or float so that you can use the appropriate syscall in main to print the median.
7. Write a function to calculate the standard deviation using the formula below. Note that there is a sqrt.s instruction in MARS. From main, save the sd and print it.

$$SD = \sqrt{\frac{\sum (r_i - r_{avg})^2}{n-1}}$$

What to turn in:

- Upload your .asm file to eLearning. If you use more than one .asm file, zip them together.

Grading Rubric:

Points	**Element**
10	Function to read integers (string) from file into a buffer
30	Function to convert string numbers to integers and store in an array
10	Function to print array of ints
40	Function for selection sort
20	Function to compute mean
30	Function to compute median
40	Function to compute standard deviation
10	Appropriate print to console, similar to sample output below
10	Comments, good use of whitespace
200	Total

Below is sample output showing the correct values for mean, median, sd.
Sample output (feel free to format your output as you like):

```
The array before:     18 9 27 5 48 16 2 53 64 98 49 82 7 17 53 38 65 71 24 31
The array after:      2 5 7 9 16 17 18 24 27 31 38 48 49 53 53 64 65 71 82 98
The mean is: 38.85
The median is: 34.5
The standard deviation is: 27.686735
-- program is finished running --
```