Assembly Project for CMPE 310

Assigned: Friday, October 9th

Due: Thursday, October 29th

Project Description:

Write an 80x86 assembly program using nasm that performs the following functions:

- Read a set of integers or floating point numbers as ASCII characters from an input file given on the command line. There will be 1 entry per line and the first line in the file will give the total number of entries in the file. There will be a maximum of 1000 entries.
- Identify the various data types in the file and output two files one with all the integers and one with all the floating point numbers. Print on screen the number of integers and floating point numbers read from the file. Print the largest and smallest floating point number on screen as well as the sum of all integers. Use the floating point unit's integer instructions to do the addition of all integers and print out the sum as a floating point number. For printing floating point numbers use %f with printf.
- Negative integers will be represented with a (-) sign at the beginning and numbers [1-9]. Positive integers will only have numbers [1-9], without a (+) sign. Convert them to integers and print them out in an output file. (project4_int.out)
- The floating point numbers will be given in standard floating point representation as, e.g. (-)9.999999E+/-99, i.e. (-) is optional, the number of digits to the right of the decimal point is variable, E is always followed by a + or a - and two digits where 09 is used for single digit exponents such as 9. Convert floating point numbers to single precision IEEE floating point representation (see the slides or text if you forgot the definition of the standard). Print out the floating points numbers to an output file. (project4_float.out)
- **NOTE:** You can **NOT** use fscanf or any other C library function to read the input file or do the conversion to floating point or integers. You must use the floating point assembly instructions to help with the conversion.
- You can still use printf or fprintf for printing from your code. You can use fopen to open a file handle for "only" the output file. Everything else in the project should be done using low level Linux System Calls (open and read).
- Make the code modular. Write subroutines to perform repetitive tasks in your project. A percentage of your grade will depend on modularity of the code. Explain every subroutine that you have written using comments in the code and also list them in your write-up along with their functional description. There are examples of these in the file mine.inc

You must use the submit program to submit your code. The class name is cmpe310 and the project name is proj4. You are also required to turn in a pdf report. Follow all the instructions given in project 1 section **Turning in your project**. Submit the project (project4.asm) file, any code that you use from our examples should be in (common_code.asm). Properly format your code using the enscript command before including in the pdf report.

You can construct your own data files for this in the format described above. We will test your code on our own examples.

THE LABS ARE INDIVIDUAL EFFORTS: INSTANCES OF CHEATING WILL RESULT IN YOU FAILING THE COURSE.