Assembly Project for CMPE 310

Assigned: Friday, Oct 5

Due: Monday, Oct 22

Project Description:

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

- Read a set of integers, strings 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. Strings will have a maximum of 256 characters.
- Identify the various data types in the file and output three files one with all the integers, one with all the strings and one with all the floating point numbers. Print on screen the number of integers, strings and floating point numbers read from the file.
- 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. (project3_int.out)
- Strings will have only characters from [a-z], [A-Z] and space. Maximum string length will be 256 characters including spaces. Print them out to an output file. (project3_string.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. (project3_float.out)
- **NOTE:** You can **NOT** use fscanf or any other C library function to do the conversion to floating point, integers or strings. 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.
- 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 proj3. You are also required to turn in a hardcopy. Follow all the instructions given in project 1 section **Turning in your project**. The breakdown of the points will also be similar as project 1. Submit the project (project3.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 printing out the hardcopy.

You can construct your own data files for this in the format described above. We will test your code on our own examples. The submitted program is due before class on Monday. You must turn in the hardcopy during the class and it must be identical to the code that you submitted.

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