Assembly Project for CMPE 310

Assigned: Thursday, Mar. 8
Due: Tuesday, March 19th (midnight)

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

Project II:

- Reads a set of floating point numbers as ASCII characters from a file, converts them to single precision IEEE floating point representation (see text for the definition of the standard), saves the floating point values into an array and outputs the sum. The 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 1.
- NOTE: You can NOT use fscanf or any other C library function to do the conversion to floating point. You may want to use the floating point assembly instructions to help with the conversion. We'll cover the floating point assembly notes this Friday.
- As before, the data file name is to be read from the command line. You are welcome to use my code examples and macros to do this project.
- You may assume the number of values in the data file never exceeds 1,000,000 elements. Therefore, you may statically declare 1 million 32-bit double words in your data segment.
- Format of the data file: Assume the file gives the number of data points on the first line. Every line following the first line contains exactly one floating point value.

Project III:

- Converts the time domain values in the floating point array that you created above into a frequency domain representation using a discrete fourier transform (DFT). I have provided you with a C code version of two routines, a DFT and a routine that converts from rectangular coordinates to polar coordinates. You need to write the nasm code that is equivalent to this code. The theoretical aspects of the DFT will be discussed in laboratory.
- The input to your program (to be read as parameters from the command line) is the file name of the time domain data (as you've done for Project II) plus the following in this order: the fundamental frequency to be used by the transform (given as a floating point number) and the number of harmonics (given as an integer). So a valid call to your assembled code looks like:

```
calc_dft y_values_file_name dft_fund_freq dft_num_freqs
calc_dft myfile.y 5.0E+7 10
```

- Your program will compute the DFT, convert to polar form and print out the frequency, magnitude and phase in three columns as floating point numbers for each frequency (starting with the DC component). You may use printf to print this table if you wish.
- Extra credit will be given for code that checks for invalid parameters, e.g. `dft_num_freqs < 2*num_y_data_points`, and/or does not use printf.

You must use the submit program to submit your code. You are also required to turn in a hardcopy. The breakdown of the points are as follows:

- Correctness 50%
- Modularity 30%
- Documentation (description, etc.) 10%
- Code Comments 10%

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 by midnight on Tuesday. You must turn in the hardcopy during class on Wednesday and it must be identical to the code that you submitted.

As always, cheating of any form will result in a zero for the project and the appropriate disciplinary action as mandated by University policy. Forms of cheating include collaboration (these projects MUST be done individually), copying code from your classmates or from the web, or the use of any automated program that converts the C code to nasm code.