Assembly Project for CMPE 310

Assigned: Friday, Feb 25

Due: Monday, Mar 7

Project Description: Hamming Distance

Write an assembly language program that prompts the user for two input strings and computes the Hamming distance between the two strings. The Hamming distance is the number of bit positions where the two strings differ. For example, the ASCII representations of the strings "foo" and "bar" in binary are:

"foo" = 0110 0110 0110 1111 0110 1111
"bar" = 0110 0010 0110 0001 0111 0010

So, the Hamming distance between "foo" and "bar" is 8.

- If the user enters two strings with different lengths, your program should return the Hamming distance up to the length of the shorter string.
- The maximum length of the string could be 255 characters.
- The easiest way to examine the contents of a register bit-by-bit is to use successive SHR instruction to shift the least significant bit into the carry flag.
- You can use the C printf function to print out the hamming distance. Example code to use printf for integer values is given on the webpage.
- The Hamming distance between the following two strings is 38:
  this is a test
  of the emergency broadcast
- You must also make your own test cases. We will test it with ours!

Turning in your program

Use the UNIX ***submit*** command on the GL system to turn in your project. You must submit the assembly language program as ***project1.asm***. The class name for submit is ***cmpe310***. The name of the assignment is ***proj1***. Check the submit help on the webpage. Due to any reason if you are going to submit your project late, the project name will be ***late***.

You are also required to turn in a hardcopy of the code and a write-up in the class. You must include a lab cover page in the hardcopy. The write-up should include the names of the various data labels and what they are used for, description of all the labels in your code, functionality of code between two labels, loops that you have used and how they are controlled etc. Most of this can also be used as comments in the code. Properly comment the code. Make your code modular. The breakdown of the points are as follows:

- Correctness 85%
- Documentation (description, etc.), code comments, modularity 15%

Properly format your code using the enscript command before printing out the hardcopy. Here are some of the useful options for enscript:

Enscript Command Help

Look at the man page for more options and details.
These options work on most linux machines. If you are on some other machine check the man page to make sure.

enscript

--columns= 2
--line-numbers
--fancy-header
--borders
--landscape
--pretty-print=asm
--output <output_file_name>.ps
--header "Header that you want"
<name of the file you want to print>

This command will produce a landscape output with 2 columns in courier size 7 font. Line numbers will be printed for each line and borders will be drawn around the columns. Long lines will be wrapped not truncated. The header on the page will be the one you give within the quotes in the --header option. It will also put at the top, the file name, date and time info, as well as page number. The pretty-print=asm option understands the keywords for asm and prints out a nice output. The output file will be the name that you give with the --output option. The last option is the asm code file that will be printed in the output file.

EXAMPLE:

e**nscript --columns=2 --line-numbers --fancy-header --borders --landscape --pretty-print=asm --output** ***project1.ps*** **--header "*Project1: Something*"** ***project1.asm***

Gives a output file project1.ps with the heading Project1: Something from the asm file project1.asm.

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