CMPE 310 Assembly Language Project I

Assigned: Friday, Feb 8

Due: Tuesday, Feb 19

Revision 1

Project Description: Checksum operation on a string of input data

Write an assembly language program that prompts the user for an input string. The program reads the input string from `stdin`, using the Linux system call `read`, into an array in memory of 128 bytes (or more). Let this be a "line". It should repeatedly continue to `read` up to the "line" buffer size until the call to `read` returns a value less than one.

This means that, to terminate the program, you can enter an empty line to end input. This can be done by pressing `CTRL+D` on a new line. This method supports UNIX piping to allow the input be an arbitrary stream (like a file).

The program should then read the data from memory in 32-bit word chunks and sum each chunk. If the number of bytes read is not a multiple of four, zeros must appear in the remainder of the last chunk. Note that the program will need to loop (after accumulation) to allow the input to be larger than the "line" buffer.

Finally, it prints out the sum in hexadecimal (see printf format string command %x). This result is useful for checking the validity of the data.

- You can use the C-library `printf` function to print out the results. An example code using `printf` for integer values is given on the webpage.

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 writeup in the class. You must include a lab cover page in the hardcopy. The writeup 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%

CMPE 310 Assembly Language Project I

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

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
--linenumbers
--fancyheader
--borders
--landscape
--prettyprint=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 prettyprint= asm option understands the keywords forasm 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:

enscript –columns= 2 –linenumbers –fancyheader –borders –landscape --prettyprint=asm --output project1.ps –header"Project1: Something" project1.asm

Gives a output file project1.ps with the heading Project1: Something from the asm fileproject1.asm.

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